Network Working Group
Internet-Draft
Intended status: Experimental
Expires: 22 November 2026

A. Jeskey
ANML Foundation

May 2026

Agentic Notation Markup Language (ANML) 1.0
draft-jeskey-anml-01

Abstract

ANML (Agentic Notation Markup Language) is a machine-first markup language for agent-to-agent and agent-to-service communication over the internet. HTML provides visual interfaces for human consumption; ANML describes content, intent, and interaction patterns optimized for machine interpretation with minimal computational overhead.

ANML defines an abstract data model that MAY be serialized as XML (application/anml+xml) or JSON (application/anml+json). Both serializations are normative and semantically equivalent. The XML serialization is recommended for human authoring and review; the JSON serialization is recommended for programmatic generation and agent-pipeline consumption.

An ANML document defines:

* Content: structured, semantic information for machine interpretation

* Interactions: operations bound to HTTP methods, endpoints, and parameters

* Knowledge: bidirectional information exchange between services and agents

* Constraints: disclosure, consent, and authorization rules that agents MUST evaluate before sharing data

* State: metadata identifying the current phase of a multi-step interaction

* Persona: advisory behavioral and tonal guidance for agent responses

ANML standardizes these elements to enable deterministic, efficient agent interactions with reduced reliance on inference over unstructured data.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

ANML is pronounced like the English word "animal":

ANML /n..ml/ AN-ih-muhl

HTML has served as the foundation of the World Wide Web since 1990, producing platform-independent hypertext documents rendered as visual interfaces for human consumption.

The emergence of large language models (LLMs) and autonomous software agents has introduced a new class of web content consumers: machines that must interpret, reason about, and act upon published information. HTML, CSS, JavaScript, images, audio, and video produce rich user experiences but encode intent, interactions, and constraints implicitly — if at all. Increased computational power does not resolve this: no amount of inference can reliably extract structured operations, disclosure rules, or workflow state from markup that was never designed to express them.

ANML addresses this gap as a machine-native markup language for agent-to-agent and agent-to-service communication. ANML complements existing web standards such as HTML and HTTP by providing a machine-oriented interaction layer, rather than replacing them. ANML is an application of XML 1.0 [XML].

The ’application/anml+xml’ Internet Media Type is defined by this specification (see Section 6.1 (Section 6.1)).

1.1. Scope

This specification defines ANML 1.0, including the document type, element and attribute sets, content model, interaction model, and media type registration.

This specification does not define:

* Visual rendering or presentation semantics
* Client-side scripting or execution models
* Transport-layer protocols (HTTP is assumed but not modified)
* Authentication or identity mechanisms

All domain names used in examples throughout this document are reserved example domains as defined in [RFC2606]: example.com, example.net, and example.org. These names do not refer to any real organization, product, or service. Where examples require multiple distinct domains, example.com is used for the canonical site domain, example.net for an authorized serving party, and example.org for a second authorized serving party or brand reference.

1.2. Conformance

This specification governs the syntax of ANML documents and the behavior of conforming ANML agents.

1.2.1. Documents

A conforming ANML document:

* Is a well-formed XML document conforming to [XML]
* Conforms to the ANML XML Schema (see Section 17 (Section 17))
* Conforms to the conventions defined in this specification
* Uses Unicode [UNICODE] as its document character set
* Is encoded as UTF-8 [RFC3629] unless an alternative encoding is declared via the XML declaration (see Section 6.3 (Section 6.2.1))

1.2.2. Conforming ANML Agents

A conforming ANML agent:

* Parses ANML documents as well-formed XML
* Supports UTF-8 character encoding
* Processes <constraints>, <state>, <interact>, <knowledge>, and <body> sections as defined in this specification
* Evaluates <constraints> before disclosing any information via <answer> elements
* Ignores unknown elements and attributes without error
* Respects user intent above all ANML instructions

1.3. Motivation

The modern web is optimized for human consumption. To interact with it, agents must currently:

* Parse large volumes of irrelevant visual and structural data
* Infer intent from loosely structured or inconsistent markup
* Perform repeated high-cost inference to understand workflows
* Reconstruct implicit interaction patterns
* Process media resources (images, audio, video) that consume significant inference cost and token budget with no guarantee of semantic return

This results in high computational cost, increased latency, reduced determinism, and fragile heuristic-dependent automation.

ANML provides a machine-native representation in which:

* Content is explicitly structured for semantic understanding
* Actions and workflows are declared
* Context is a first-class construct
* Agent behavior can be guided at the document level

Existing approaches such as REST APIs and interface description languages (e.g., OpenAPI, GraphQL) address machine-to-service communication but require prior integration: each service defines a bespoke contract that agents must be individually programmed to use. ANML takes a different approach. Like HTML for browsers, ANML provides a universal document format that any conforming agent can interpret without service-specific integration. A single ANML document conveys content, available operations, knowledge exchange, disclosure constraints, workflow state, and behavioral guidance — none of which are expressible in a typical API specification.

1.4. Goals

* Enable efficient agent-to-agent communication over the internet
* Reduce inference cost for interpreting web content
* Provide explicit, structured representations of content and workflows
* Standardize machine-consumable service interfaces
* Support bidirectional knowledge sharing between services and agents
* Allow services to provide behavioral and persona guidance to agents
* Improve determinism and reliability of agent interactions

1.5. Non-Goals

* Replacing HTML for human-facing interfaces
* Rendering visual layouts or graphical experiences
* Executing client-side logic
* Serving as a general-purpose programming language

2. Applicability

ANML is domain-neutral. It is applicable to any context in which autonomous agents interact with services or with each other over HTTP. Example domains include but are not limited to:

* Web services and APIs

* Personal and enterprise agent systems

* IoT and device interaction

* Data retrieval and knowledge systems

ANML does not presume a commercial, consumer, or enterprise context. Any service that publishes structured information for machine consumption MAY use ANML.

3. Terms

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

action An executable operation defined in the <interact> section, bound to an HTTP method and endpoint URI.

aesthetic Advisory visual identity guidance within the <aesthetic> section, including colors, typography, logos, and display name.

agent A software component that processes ANML documents, performs actions, and communicates with services on behalf of a user or another agent. Also referred to as "user agent".

ANML document A well-formed XML document conforming to this specification and the ANML XML Schema (see Section 17 (Section 17)).

answer A response element providing information requested by an <ask> element. Answers may flow from agent to service or from service to agent.

ask A declarative element through which one party requests information from the other. Neither party is required to fulfill any ask.

body The primary structured content of an ANML document, contained within the <body> element.

confidentiality An attribute on <inform> governing how an agent may share service-provided information with other services.

conforming ANML agent An agent that conforms to this specification in its processing of the ’application/anml+xml’ media type.

constraint A rule governing whether and how requested information may be disclosed by an agent.

data A structured collection of items within the <body>, represented by the <data> element.

description A textual description of a media resource, provided as a child of <img>, <audio>, or <video>. Enables agents to understand media content without performing inference.

disclosure The act of an agent sharing information with a service in response to an <ask>, subject to applicable constraints and user consent.

endpoint A URI to which an action is directed.

flow A declaration of the structure of a multi-step workflow within the <state> section.

inform A knowledge element through which a service communicates information to an agent.

knowledge The section containing <inform> and <ask> elements, enabling bidirectional information exchange.

media type An Internet Media Type as defined in [RFC6838].

model An advisory preference within <persona> indicating which inference model or capability a service recommends for processing the document.

nav A pagination and continuation element within <body>, providing next, prev, and cursor references for multi-page result sets.

persona Advisory behavioral and tonal guidance within the <persona> section.

refuse A response element explicitly declining to fulfill an <ask>.

rights A copyright and licensing declaration within the <rights> element in <footer>.

service A networked application that publishes ANML documents and processes agent responses.

state The section carrying contextual information about a multi-step interaction.

status An element communicating the outcome of a preceding operation, including success, error, or partial completion.

step A single stage within a workflow.

transcript A textual transcript of an <audio> or <video> resource, provided as a child element. Enables agents to understand media content without performing inference.

serving domain The registered domain (eTLD+1) of the URL from which an ANML document was retrieved, normalized to lowercase ASCII, with port components omitted. For example, an ANML document retrieved from 'https://cdn.example.net:443/anml' has a serving domain of 'example.net'. The serving domain is the value used as the first element of the trust cache key and as the domain parameter in Trust Query requests. When the URL origin and the TLS certificate subject differ, the URL origin takes precedence. Agents SHOULD use the Public Suffix List to determine eTLD+1 boundaries.

site domain The canonical domain of the site whose identity is asserted in an ANML document, as specified in the domain attribute of a <trust> or <site> element. Distinct from the serving domain when a third party is serving content on behalf of the site.

URI A Uniform Resource Identifier as defined in [RFC3986].

usage A machine-readable declaration governing what an agent may do with content. Values form a hierarchy: none < display < cache < store < train.

well-formed XML An XML document satisfying the well-formedness constraints defined in [XML].

4. Design Principles

The following principles guide the design of ANML and SHOULD inform the behavior of conforming agents and services.

Machine-First. ANML targets machine consumption, not human presentation.

Explicit Over Implicit. Interactions SHOULD be explicitly defined. Agents SHOULD NOT need to infer intent from ambiguous markup.

Minimize Inference Cost. Documents SHOULD be structured for direct interpretation, minimizing the need for inference.

Separation of Concerns. Each section of an ANML document has a distinct responsibility.

Privacy by Design. Agents control disclosure. Services MAY request information but MUST NOT require it.

Deterministic Behavior. Agents SHOULD produce consistent results given identical inputs.

Progressive Enhancement. ANML MAY be adopted incrementally alongside existing web infrastructure.

Domain Neutrality. ANML is domain-agnostic. It is not limited to commercial or e-commerce contexts and applies equally to any agent-to-agent or agent-to-service interaction.

Extensibility. Conforming agents MUST safely ignore unknown elements and attributes.

Trust Awareness. Agents SHOULD evaluate service trustworthiness before disclosing information or executing actions.

Human Intent First. User intent overrides all ANML instructions. An agent MUST NOT act contrary to the expressed wishes of its user.

5. ANML as an Application of XML

ANML is an application of XML 1.0 [XML] and XML Namespaces [XMLNS]. This specification assumes familiarity with XML.

5.1. XML Documents

An ANML document is a well-formed XML document consisting of a hierarchy of elements as defined in Section 8.

The ANML namespace URI is:

urn:ietf:params:xml:ns:anml:1.0

A conforming ANML document MUST declare this namespace on the root element:

```
<anml xmlns="urn:ietf:params:xml:ns:anml:1.0">
  ...
</anml>
```

5.2. ANML Lexical Syntax

ANML inherits its lexical syntax from XML 1.0. The following subsections define additional constraints.

5.2.1. Element and Attribute Names

All element and attribute names defined by this specification are lowercase. Conforming documents MUST use lowercase names for all elements defined herein.

5.2.2. Attribute Values

Attribute values MUST be enclosed in double quotes (U+0022). All attribute values are treated as strings. ANML does not define implicit type coercion.

5.2.3. Whitespace

Whitespace within element content is significant and MUST be preserved by conforming agents. ANML does not define whitespace collapsing rules.

5.2.4. Character References

Numeric character references and the predefined XML entity references are supported as defined in [XML]. ANML does not define additional named entity references.

5.2.5. CDATA Sections

CDATA sections MUST NOT appear in conforming ANML documents.

5.2.6. Comments and Processing Instructions

XML comments MAY appear in ANML documents. Conforming agents MAY ignore comments; comments MUST NOT carry semantic meaning. Processing instructions other than the XML declaration MUST NOT appear in conforming ANML documents.

5.2.7. Unknown Elements and Attributes

Conforming agents MUST ignore any element or attribute not defined by this specification or a recognized extension namespace, without raising an error.

5.3. ANML Namespace and Public Identifiers

5.3.1. Namespace

ANML uses XML Namespaces [XMLNS] to distinguish its elements from those of other vocabularies. The namespace URI 'urn:ietf:params:xml:ns:anml:1.0' is an identifier and need not resolve to a retrievable resource.

5.3.2. Namespace Policy

The ANML namespace URI 'urn:ietf:params:xml:ns:anml:1.0' is version-specific — the '1.0' is part of the URI. This means a future ANML 2.0 specification using a different namespace URI would be treated as an entirely distinct namespace by conforming XML processors. Documents from the two versions would not be interoperable at the namespace level.

The author has noted the alternative approach used by HTML5 and other standards: a version-independent namespace URI (e.g., 'urn:ietf:params:xml:ns:anml') with the version carried only in the version attribute of the root element. This approach avoids namespace fragmentation across versions and is RECOMMENDED for consideration before the namespace URI is formally registered with IANA.

NOTE: the choice between a version-specific and version-independent namespace URI should be resolved before standards-track publication. The current version-specific URI is used throughout this draft pending that decision. If a version-independent URI is adopted, all namespace URI references in this document will be updated accordingly.

5.3.3. Versioning

The namespace URI includes a version component ('1.0'). Future major versions will use a new namespace URI. The namespace URI is the authoritative version indicator.

5.3.4. Extension Namespaces

ANML documents MAY include elements and attributes from other XML namespaces. Extensions MUST be namespace-qualified and MUST NOT use the ANML namespace URI.

5.3.5. Document Type Declaration

ANML documents SHOULD NOT include a Document Type Declaration (DOCTYPE). If present, conforming agents MUST NOT process its internal subset or resolve external entities referenced by it.

5.4. Example ANML Document

```
<?xml version="1.0" encoding="UTF-8"?>
<anml xmlns="urn:ietf:params:xml:ns:anml:1.0" ttl="3600">
  <head>
    <title>Travel Booking Service</title>
    <meta name="type" value="service"/>
  </head>
  <constraints>
    <disclosure field="airline" requires="explicit-consent"/>
  </constraints>
  <state>
    <context><step>search</step></context>
    <flow>
      <step id="search" label="Search flights" status="current"/>
      <step id="select" label="Select a flight" status="pending"/>
      <step id="payment" label="Payment"
          status="pending" required="true"/>
      <step id="confirm" label="Confirmation" status="pending"/>
    </flow>
  </state>
  <interact>
    <action id="submit-airline" method="POST" endpoint="/airline"/>
  </interact>
  <knowledge>
    <inform ttl="3600">
      We offer flights to over 200 destinations worldwide.
    </inform>
    <ask field="airline" action="submit-airline"
         required="false" purpose="personalization"/>
  </knowledge>
  <persona>
    <model capability="reasoning"/>
    <language policy="native"/>
    <tone value="friendly"/>
    <instructions>Be helpful and concise.</instructions>
  </persona>
  <body>
    Book flights to your destination.
  </body>
  <footer>
    <rights holder="Example Travel, Inc." year="2026" usage="cache">
      Copyright 2026 Example Travel, Inc.
    </rights>
  </footer>
</anml>
```

6. ANML as an Internet Media Type

6.1. application/anml+xml Media Type

The following is to be registered with IANA:

Media Type name: application

Media subtype name: anml+xml

Required parameters: none

Optional parameters: charset (default UTF-8), version (default "1.0")

Encoding considerations: Same as for "application/xml" as specified in [XMLMIME].

Security considerations: See Section 13 (Section 13).

Interoperability considerations: Conforming agents MUST safely ignore unknown elements and attributes.

Published specification: This document.

Applications that use this media type: Autonomous software agents, agent-to-agent communication systems.

File extension(s): .anml

6.2. ANML Document Representation

6.2.1. Character Encoding

The default character encoding for ANML documents is UTF-8 [RFC3629]. When transmitted via HTTP, the charset parameter of the Content-Type header takes precedence over the XML declaration, as specified in [XMLMIME].

6.2.2. XML Declaration

A conforming ANML document SHOULD begin with an XML declaration. The XML declaration is REQUIRED when the document uses an encoding other than UTF-8.

6.2.3. Byte Order Mark

UTF-8 encoded documents SHOULD NOT begin with a BOM. Conforming agents MUST accept documents that include one. For UTF-16, the BOM is REQUIRED.

6.2.4. Line Endings

ANML documents SHOULD use LF (U+000A). Conforming agents MUST accept LF, CR, or CR+LF and normalize to LF as specified in [XML] Section 2.11.

6.3. Content Negotiation

Content negotiation follows standard HTTP mechanisms as defined in [RFC9110]. An agent SHOULD include 'application/anml+xml' in the Accept header. A service MUST set the Content-Type header when responding with an ANML document.

6.4. HTTP Caching Interaction

The interaction between ANML's ttl attribute and HTTP caching directives is defined in Section 8.1.1, adjacent to the ttl attribute definition on the <anml> root element. It is cross-referenced here because HTTP caching is a transport-layer concern. The normative content is in Section 8.1.1; this section serves as a locator for readers approaching from the transport layer.

6.5. Discovery

6.5.1. Well-Known URI

A service MAY publish a discovery document at '/.well-known/anml' [RFC8615]. The serving domain of the well-known URI response is implicitly authoritative: an agent MUST treat an ANML document retrieved from 'https://example.net/.well-known/anml' as asserting the identity of example.net, regardless of any title, persona, or display-name content in the document. A <trust> element or <site> element claiming a different domain in a well-known URI response MUST be treated as a third-party assertion subject to the trust verification procedure in Section 12 — it does not override the serving domain's implicit authority over its own well-known URI. This rule prevents CDN operators or hosting providers from substituting their own ANML identity for a served site's identity at the discovery endpoint. The following normative rules govern agent behavior when interacting with this URI:

* A 404 (Not Found) response at '/.well-known/anml' SHOULD be interpreted by agents as a signal that the service does not currently support ANML. Agents SHOULD NOT retry this URI within the same session. Agents MAY retry in a new session or after the HTTP cache TTL of the 404 response has expired. A 404 is not treated as permanent — a service may deploy ANML after a 404 has been cached, and agents MUST be capable of rediscovering ANML support across sessions.

* A 301 (Moved Permanently) or 308 (Permanent Redirect) response SHOULD be followed. Agents SHOULD update their cached discovery URI to the redirect target. Temporary redirects (302, 307) MAY be followed but MUST NOT update the cached discovery URI.

* A service that supports ANML but has no content to serve at '/.well-known/anml' SHOULD return a minimal valid ANML document rather than a 404. A minimal valid document contains at minimum a <head> element with a <title>. This allows agents to confirm ANML support without inferring from a 404 that support is absent.

* A 410 (Gone) response MUST be interpreted as a permanent removal of ANML support. Agents MUST NOT retry this URI.

6.5.2. HTTP Link Header

A service MAY advertise ANML support via a Link header with rel="alternate" and type="application/anml+xml".

6.5.3. HTML Link Element

A service MAY include a <link rel="alternate" type="application/anml+xml"> element in HTML documents.

6.5.4. DNS Service Discovery

A domain MAY publish SRV records [RFC2782] at '_anml._tcp' and TXT records with key 'v=anml1' and optional 'path' key.

7. ANML JSON Serialization

ANML defines an abstract data model that MAY be serialized as either XML (Section 5) or JSON. Both serializations are normative and semantically equivalent. An implementation that correctly processes one serialization MUST produce results identical to processing the other serialization of the same document. The serialization format is a deployment choice, not a semantic choice.

The JSON serialization is identified by the media type ’application/anml+json’. Content negotiation follows standard HTTP mechanisms: a client SHOULD include both ’application/anml+xml’ and ’application/anml+json’ in the Accept header with appropriate q-values to indicate preference. A service MUST set the Content-Type header to the serialization actually returned.

The XML serialization (Section 5) is RECOMMENDED for documents intended for human authoring and review. The JSON serialization is RECOMMENDED for programmatic generation and agent-pipeline consumption. Both are REQUIRED to be supported by conforming implementations claiming full conformance. Implementations MAY support only one serialization and MUST document which serializations they support.

7.1. application/anml+json Media Type

The following is to be registered with IANA:

Media Type name: application

Media subtype name: anml+json

Required parameters: none

Optional parameters: version (default "1.0")

Encoding considerations: JSON text MUST be encoded as UTF-8 as specified in RFC 8259. A BOM MUST NOT be prepended.

Security considerations: See Section 11.

Interoperability considerations: Conforming agents MUST safely ignore unknown keys.

Published specification: This document.

Applications that use this media type: Autonomous software agents, agent-to-agent communication systems, REST API clients.

File extension(s): .anml.json

7.2. General Mapping Rules

The following rules govern the translation between the XML and JSON serializations of ANML. These rules are applied mechanically and uniformly across all elements.

7.2.1. Root Object

An ANML JSON document is a single JSON object. The root object corresponds to the <anml> element. XML attributes of <anml> become top-level keys of the root object. The version key is REQUIRED. Example:

```
{
  "anml": "1.0",
  "lang": "en",
  "ttl": 3600,
  "head": { ... },
  "constraints": { ... },
  "state": { ... },
  "interact": { ... },
  "knowledge": { ... },
  "persona": { ... },
  "aesthetic": { ... },
  "body": { ... },
  "footer": { ... },
  "status": { ... }
}
```

7.2.2. Attributes to Properties

XML attributes become JSON string, number, or boolean properties of the enclosing object. Attribute names that contain hyphens (e.g., supported-versions) are preserved as-is in JSON keys. Boolean attributes with values "true" or "false" become JSON boolean values true or false. Numeric attributes (ttl, min, max) become JSON numbers. All other attributes become JSON strings.

7.2.3. Text Content

XML element text content (#PCDATA) becomes a JSON string value associated with the key "content" within the element’s object representation. When an element has ONLY text content and no attributes or child elements, its value MAY be represented in compact form as a JSON string rather than an object with a "content" key. Example:

```
XML:  <inform ttl="3600">We offer 200 destinations.</inform>

JSON (full):
{
  "ttl": 3600,
  "content": "We offer 200 destinations."
}

JSON (shorthand, only when no other attributes or children):
"We offer 200 destinations."
```

7.2.4. Child Elements to Nested Objects

XML child elements become nested JSON objects keyed by the element name. Elements that may appear multiple times as siblings (repeatable elements) MUST always be represented as a JSON array, even when only one instance is present. This rule is absolute: a repeatable element is never a bare object — it is always a single-element array or a multi-element array. Elements that may appear at most once (non-repeatable elements) are represented as a bare JSON object, never as an array.

Repeatable elements (MUST use array form): inform, ask, answer, refuse, action, param, option, step, meta, logo, color, font, section, item, field, attribution.

Non-repeatable elements (MUST use object form): head, constraints, state, interact, knowledge, persona, aesthetic, body, footer, status, context, flow, response, model, language, tone, voice, instructions, vocabulary, rights, nav, display-name, colors, typography, title, trust.

Example:

```
XML:
<knowledge>
  <inform ttl="3600">Fact one.</inform>
  <inform ttl="7200">Fact two.</inform>
  <ask field="name" action="submit" required="false"/>
</knowledge>

JSON:
{
  "knowledge": {
    "inform": [
      { "ttl": 3600, "content": "Fact one." },
      { "ttl": 7200, "content": "Fact two." }
    ],
    "ask": {
      "field": "name",
      "action": "submit",
      "required": false
    }
  }
}
```

7.2.5. Empty Elements

XML elements with no content and no child elements (EMPTY content model) become JSON objects containing only their attribute properties. An empty element with no attributes becomes an empty JSON object {}. Example:

```
XML:  <tone value="friendly"/>
JSON: { "tone": { "value": "friendly" } }

XML:  <disclosure field="email" requires="explicit-consent"/>
JSON: { "disclosure": { "field": "email",
                        "requires": "explicit-consent" } }
```

7.2.6. Unknown Keys

Conforming ANML agents processing JSON documents MUST ignore any key not defined by this specification or a recognized extension, without raising an error. This mirrors the requirement for XML unknown elements and attributes in Section 5.2.7.

7.3. Complete JSON Example

The following is the JSON serialization of the example ANML document in Section 5.4:

```
{
  "anml": "1.0",
  "ttl": 3600,
  "head": {
    "title": "Travel Booking Service",
    "meta": [
      { "name": "type", "value": "service" }
    ]
  },
  "constraints": {
    "disclosure": [
      { "field": "airline", "requires": "explicit-consent" }
    ]
  },
  "state": {
    "context": { "step": "search" },
    "flow": {
      "step": [
        { "id": "search",  "label": "Search flights",
          "status": "current" },
        { "id": "select",  "label": "Select a flight",
          "status": "pending" },
        { "id": "payment", "label": "Payment",
          "status": "pending", "required": true },
        { "id": "confirm", "label": "Confirmation",
          "status": "pending" }
      ]
    }
  },
  "interact": {
    "action": [
      {
        "id": "submit-airline",
        "method": "POST",
        "endpoint": "/airline"
      }
    ]
  },
  "knowledge": {
    "inform": { "ttl": 3600, "content": "We offer flights to over 200 destinations worldw
ide." },
    "ask": {
      "field": "airline",
      "action": "submit-airline",
      "required": false,
      "purpose": "personalization"
    }
  },
  "persona": {
```

```
    "model": { "capability": "reasoning" },
    "language": { "policy": "native" },
    "tone": { "value": "friendly" },
    "instructions": "Be helpful and concise."
  },
  "body": {
    "content": "Book flights to your destination."
  },
  "footer": {
    "rights": {
      "holder": "Example Travel, Inc.",
      "year": "2026",
      "usage": "cache",
      "content": "Copyright 2026 Example Travel, Inc."
    }
  }
}
```

7.4. Discovery and Content Negotiation

A service supporting JSON serialization SHOULD include 'application/anml+json' in the Accept-Patch or Alternates header of its well-known URI response. Agents SHOULD include both media types in the Accept header of discovery requests:

```
Accept: application/anml+xml;q=0.9, application/anml+json;q=1.0
```

A service MAY serve different serializations from different endpoints or from the same endpoint based on content negotiation. The data model MUST be identical regardless of serialization. A service MUST NOT include information in one serialization that is absent from the other.

7.5. JSON-Specific Security Considerations

JSON documents are subject to the general security considerations in Section 11. The following additional considerations apply specifically to JSON serialization:

* JSON parsers MUST impose limits on nesting depth to prevent stack exhaustion from deeply nested objects. The RECOMMENDED maximum nesting depth is 32 levels.

* JSON parsers MUST impose limits on document size. The RECOMMENDED maximum document size is 1MB.

* JSON parsers MUST reject duplicate keys within the same object. When duplicate keys are present the document MUST be treated as malformed and rejected entirely. Note: the XML serialization provides equivalent protection through XML 1.0 Section 2.3 well-formedness constraints, which prohibit duplicate attribute names — a conforming XML parser will reject such documents before they reach ANML processing. Both serializations therefore enforce the same uniqueness invariant through their respective base specifications.

* JSON numbers MUST be parsed as IEEE 754 double-precision floating-point values. Implementations MUST handle numeric overflow and precision loss gracefully.

* JSON strings MUST be valid UTF-8. Implementations MUST reject documents containing invalid UTF-8 sequences.

7.6. Normative Equivalence

This section is normative. The following assertions MUST hold for any conforming implementation processing the XML and JSON serializations of the same logical ANML document:

* Disclosure decisions: the agent MUST make identical disclosure decisions — which <ask> fields to answer, refuse, or defer — regardless of whether the document was received as XML or JSON.

* Action execution: the agent MUST identify the same set of executable actions and apply the same confirm, auth, and idempotent constraints regardless of serialization.

* Knowledge exchange responses: the agent MUST produce semantically equivalent <answer>, <refuse>, and <inform> responses regardless of serialization. The response itself MAY use either serialization independently of the received serialization.

* Trust tier determinations: the agent MUST assign the same trust tier to each domain regardless of serialization.

* Constraint evaluation: the agent MUST apply the same disclosure constraints regardless of serialization.

* Usage enforcement: the agent MUST apply the same usage restrictions to content regardless of serialization.

The following are explicitly NOT required to be identical across serializations: user interface rendering, caching implementation details, logging format, and internal representation. These are implementation-defined and cannot be specified by this document.

Any ANML document expressible in XML serialization MUST be expressible in JSON serialization with equivalent semantics, and vice versa. Where ambiguity exists between the XML and JSON mapping rules, the XML serialization is authoritative. Implementers encountering ambiguity SHOULD file an issue with the ANML Foundation at spec@anmlfoundation.org.

8. Document Structure

An ANML document is a tree of elements rooted at <anml>, organized into sections with distinct responsibilities.

8.1. Root Element: anml

The <anml> element is the root element. Attributes: xmlns (REQUIRED), version (OPTIONAL, default "1.0"), role (OPTIONAL, "service"|"agent-response" — declares the document type, enabling schema-based validation of role-specific constraints. Conforming implementations MUST generate the role attribute on all documents they produce. When the role attribute is absent, agents MUST infer document role from HTTP context: a document received in response to an HTTP GET at a well-known URI or HTTP Link header discovery endpoint is a service document; a document submitted as the body of an HTTP POST to an action endpoint is an agent-response document. When present, the role attribute value MUST match the inferred HTTP context role — if they conflict the document MUST be rejected. Validators MUST apply role-appropriate content model rules when this attribute is present. There is no default value — absence triggers context inference, not a default assumption), supported-versions (OPTIONAL, comma-separated list of supported version strings), ttl (OPTIONAL, non-negative integer, seconds until the document SHOULD be considered stale, counted from the value of the HTTP Date response header at time of retrieval; if no Date header is present, ttl is counted from time of receipt; if ttl conflicts with HTTP Cache-Control, the interaction is governed by Section 6.5 (HTTP Caching Interaction); see also Section 8.1.1 for the caching subsection adjacent to this attribute definition), lang (OPTIONAL, BCP 47).

Content model (single-site document): head?, constraints?, state?, interact?, knowledge?, persona?, aesthetic?, body?, footer?, status?

Content model (multi-site document): site+

A conforming <anml> document MUST use one of these two content models exclusively. An <anml> element MUST NOT mix top-level identity sections (persona, aesthetic, knowledge, etc.) with <site> elements. When a document asserts identity for more than one serving party, it MUST use the multi-site content model with <site> wrappers for all parties including the serving domain itself.

Version negotiation: when an agent encounters a document whose version attribute specifies a version the agent does not implement, the agent SHOULD attempt best-effort processing, ignoring elements and attributes it does not recognize, and SHOULD inform the user that the document uses a version newer than the agent supports. If the supported-versions attribute is present and does not include any version the agent implements, the agent MAY refuse to process the document and MUST NOT execute any actions defined within it. The agent SHOULD surface a meaningful error to the user in this case. This behavior mirrors established web precedent: agents degrade gracefully rather than failing completely when encountering newer versions.

The RECOMMENDED authoring order for single-site documents places framing sections (constraints, state, interact, knowledge, persona) before <body> to optimize for streaming consumption.

8.1.1. Interaction with HTTP Caching

The ANML ttl attribute interacts with HTTP caching headers as follows, in order of precedence:

* If the HTTP response carries Cache-Control: no-store, the agent MUST NOT cache the document or any content derived from it, regardless of any ttl attribute value. no-store takes absolute precedence over ttl.

* If the HTTP response carries Cache-Control: no-cache, the agent MUST revalidate the document with the server before each use, regardless of any ttl attribute value.

* If the HTTP response carries Cache-Control: max-age=N, and the document carries ttl=M, the effective TTL is min(N, M) — the more restrictive value applies.

* If Cache-Control: s-maxage is present, it governs shared cache behavior per RFC 9111 Section 5.2.2.6 independently of ANML ttl semantics, which apply to agent-side caching only.

* If Cache-Control: stale-while-revalidate=N is present, agents MAY serve stale ANML content for up to N seconds while revalidating in the background, provided the content has not exceeded its ANML ttl.

* In all cases not addressed above, RFC 9111 [RFC9111] governs HTTP caching behavior and ANML ttl provides an additional upper bound on agent-side content freshness.

8.2. Site Element: site

The <site> element is a wrapper used in multi-site documents to group identity sections belonging to a single serving party or trusted third party. It replaces individual top-level identity sections when a document asserts identity for more than one domain.

Attributes:

domain The canonical domain of the site whose identity this element asserts. REQUIRED. When domain matches the serving domain, no trust verification is required. When domain differs from the serving domain, the agent MUST verify trust delegation as defined in Section 12 before processing the contents of this <site> element.

trust-verified ISO 8601 datetime at which the serving party last verified the _anml DNS record and authorization data for this domain. OPTIONAL. Agents MAY use this to decide whether to revalidate a cached authorization: if trust-verified is more recent than the agent's cached result, the agent SHOULD revalidate. Agents MUST NOT treat this as authorization to skip independent verification — it is a hint only. When this attribute is absent, agents MUST perform verification from the DNS bootstrap record as defined in Section 12.4.

Content model: head?, constraints?, state?, interact?, knowledge?, persona?, aesthetic?, body?, footer?, status?, site-ref*

When the domain attribute differs from the serving domain, the agent MUST perform trust verification by looking up the _anml DNS TXT record for the site domain and proceeding per Section 12.4. No additional trust-manifest attribute is required — the domain attribute alone triggers the verification procedure.

A <site> element MUST contain at least one child element. An empty <site> element is malformed. Each <site> element in a document MUST have a unique domain attribute value. Duplicate domain values within the same <anml> element are not permitted.

8.3. Head Section: head

The <head> element contains document metadata. Content model: title?, meta*.

8.3.1. Title: title

The <title> element provides the document title. Content model: (#PCDATA).

8.3.2. Meta: meta

The <meta> element provides name/value metadata pairs. Attributes: name (identifier string), value (string). Content model: EMPTY.

8.4. Constraints Section: constraints

The <constraints> element defines rules governing disclosure. Agents MUST evaluate constraints before submitting any <answer>. Content model: disclosure*.

8.4.1. Disclosure: disclosure

The <disclosure> element defines a single disclosure rule. Attributes: field (REQUIRED, references an <ask> field), requires (REQUIRED: explicit-consent|implicit-consent|authentication|none). Content model: EMPTY.

8.5. State Section: state

The <state> element carries contextual information about multi-step interactions. Content model: context?, flow?.

8.5.1. Context: context

The <context> element identifies the current position within a flow. Content model: step. The child <step> element contains (#PCDATA) whose text MUST match a step id defined in <flow>. This <step> child carries no attributes and serves solely as a reference to a flow step.

8.5.2. Flow: flow

The <flow> element declares the workflow structure. Content model: step*.

8.5.3. Step: step (flow)

The <step> element defines a single workflow stage. Attributes: id (REQUIRED), label, status (completed|current|pending|skipped), required (boolean, "true"|"false", default "false"), next (step id of the subsequent step), condition (a service-defined opaque string evaluated by the service to determine step availability; agents MUST treat this as informational and MUST NOT attempt to evaluate condition expressions independently), action (references an <action> id available at this step). Content model: EMPTY.

8.6. Interaction Section: interact

The <interact> element defines executable operations. Content model: action*.

8.6.1. Action: action

The <action> element defines a single operation. Attributes: id (REQUIRED), method (REQUIRED, HTTP method), endpoint (REQUIRED, URI), enctype (media type for request body, default "application/x-www-form-urlencoded"), auth (none|required|optional, default "none"), idempotent (boolean, "true"|"false"), confirm (boolean, "true"|"false", if "true" the agent SHOULD confirm with the user before executing), description (human-readable text). Content model: param*, response?.

8.6.2. Param: param

The <param> element declares an action parameter. Attributes: name, type (string|number|boolean|date|datetime|uri|enum), required (boolean, "true"|"false", default "false"), default, description, pattern, min, max. Content model: option*.

8.6.3. Option: option

The <option> element defines a permitted value for enum parameters. Attributes: value (REQUIRED, string), label (human-readable display text). Content model: EMPTY.

8.6.4. Response: response

The <response> element describes the expected response. Attributes: type (default application/anml+xml), description. Content model: EMPTY.

8.7. Knowledge Section: knowledge

The <knowledge> element defines bidirectional information exchange. Content model: (inform | ask)* for service documents; (answer | refuse | ask | inform)* for agent responses.

8.7.1. Inform: inform

The <inform> element communicates information from service to agent. Attributes: ttl (non-negative integer, seconds from HTTP Date header of the response in which this element was received; see Section 8.1 for ttl semantics), scope (element id this inform applies to), priority (low|normal|high, default "normal"), confidentiality (public|restricted|private, default "public"), usage (none|display|cache|store|train — see Section 15.4 for the normative usage hierarchy). Content model: (#PCDATA).

8.7.2. Ask: ask

The <ask> element requests information from the other party. Attributes: field (REQUIRED), action (REQUIRED, references an <action> id), required (boolean, "true"|"false", default "false"), purpose (human-readable reason for the request), type (expected value type: string|number|boolean|date|datetime|uri — see Section 8.14 for the unified ANML type system definition). Content model: EMPTY.

The field attribute is a free-form string identifying the information being requested. For interoperability, field values representing personal information SHOULD use the property names defined in vCard version 4.0 [RFC6350] where applicable — for example: "fn" (formatted name), "email", "tel", "adr", "bday", "gender", "lang". Use of these standard property names allows agents to recognize equivalent requests across services that use the same vocabulary. Service-specific field names are permitted and are opaque to agents that do not recognize them; agents MUST NOT refuse to process an <ask> solely because the field name is unrecognized — they MUST instead evaluate it against applicable constraints and user consent as with any other field. When an agent encounters an <ask> with an unrecognized field name, it MUST apply the most restrictive applicable constraint as a default. Specifically: if no <disclosure> constraint is specified for the unrecognized field, the agent MUST treat it as requiring requires="explicit-consent" regardless of any other constraint configuration. This ensures unrecognized fields are never disclosed silently. An IANA registry of standard ANML field names is planned for a future revision.

8.7.3. Answer: answer

The <answer> element provides requested information. Attributes: field (REQUIRED), value (REQUIRED), consent (explicit|implicit|delegated), consent-granted (OPTIONAL, ISO 8601 datetime in UTC — the time at which the user granted consent for this disclosure; allows services to evaluate whether consent is current relative to their valid-for requirement). Content model: EMPTY.

Services that specify valid-for on a <disclosure> element SHOULD evaluate the consent-granted timestamp in the corresponding <answer> to determine whether the agent's consent is still valid. If consent-granted is absent or older than valid-for seconds, the service SHOULD treat the disclosure as requiring fresh consent and SHOULD return a response requesting re-consent rather than processing the stale answer.

8.7.4. Refuse: refuse

The <refuse> element explicitly declines an <ask>. Attributes: field (REQUIRED), reason (REQUIRED: constraint-violation|user-denied|policy-violation|unsupported-field|trust-insufficient), constraint (references a <disclosure> field, if applicable), message (human-readable explanation). Content model: EMPTY.

8.8. Persona Section: persona

The <persona> element provides advisory guidance on agent communication style. Content model: model?, language?, tone?, voice?, instructions?, vocabulary?.

Precedence: agents SHOULD apply persona guidance to their responses. However, user safety policies, user-configured preferences, and platform-level guardrails take absolute precedence over all persona guidance. The SHOULD in this section applies only to persona guidance that does not conflict with any higher-priority constraint. When persona guidance conflicts with a safety policy or user preference, the MUST NOT in Section 13.3 governs — the SHOULD creates no obligation to apply conflicting guidance. Agents MUST be able to explain to a user, on request, which persona guidance was applied and which was overridden.

8.8.1. Model: model

The <model> element provides an advisory model preference for agent inference. Agents MAY use this guidance to select an appropriate model but are not required to honor it. Attributes: name (model identifier, e.g., "claude-4", "gpt-5"), provider (e.g., "anthropic", "openai"), capability (e.g., "vision", "code", "reasoning"). All attributes are OPTIONAL. When name is not available, agents SHOULD select a model that satisfies the specified capability. Content model: EMPTY.

8.8.2. Language: language

The <language> element specifies the response language policy. Attributes: value (BCP 47 tag, used when policy="fixed"), policy (native|match|fixed). Content model: EMPTY.

Policy values: "native" — the agent SHOULD respond in the user's preferred language as known to the agent; "match" — the agent SHOULD respond in the language of the document; "fixed" — the agent MUST respond in the language specified by the value attribute. Tone and voice values are advisory opaque strings; interoperability of persona guidance across services is not guaranteed and agents SHOULD treat unrecognized tone and voice values as informational only.

Privacy note: policy="native" causes the agent to infer or disclose the user's preferred language to influence its response. The user's language preference is personal information. Agents SHOULD treat the application of policy="native" as equivalent to an implicit disclosure of a language preference field and SHOULD apply any applicable <constraints> governing language-related disclosures. Services MUST NOT use policy="native" as a mechanism to fingerprint users by language.

8.8.3. Tone: tone

The <tone> element specifies the emotional register. Attributes: value (e.g., "friendly", "formal", "neutral"). Content model: EMPTY.

8.8.4. Voice: voice

The <voice> element specifies character and perspective. Attributes: perspective (first|third), name (character name, string). Content model: EMPTY.

8.8.5. Instructions: instructions

The <instructions> element provides free-text behavioral guidance. Advisory only. Content model: (#PCDATA).

8.8.6. Vocabulary: vocabulary

The <vocabulary> element specifies preferred and avoided terms. Content model: prefer*, avoid*.

8.8.7. Prefer: prefer

The <prefer> element declares a preferred term or phrase. Content model: (#PCDATA).

8.8.8. Avoid: avoid

The <avoid> element declares a term or phrase that SHOULD NOT be used. Content model: (#PCDATA).

8.9. Aesthetic Section: aesthetic

The <aesthetic> element provides advisory visual identity guidance. Content model: display-name?, logo*, colors?, typography?.

8.9.1. Display Name: display-name

The <display-name> element specifies the preferred service name. Content model: (#PCDATA).

8.9.2. Logo: logo

The <logo> element references a logo image. Attributes: src (URI), alt (text description), type (media type, e.g., "image/svg+xml"), variant (e.g., "icon", "wordmark", "full"). Content model: EMPTY.

8.9.3. Colors: colors

The <colors> element defines the service color palette. Content model: color*.

8.9.4. Color: color

The <color> element defines a single color value. Attributes: role (e.g., primary, secondary, accent), value (CSS color string). Content model: EMPTY.

8.9.5. Typography: typography

The <typography> element defines font preferences. Content model: font*.

8.9.6. Font: font

The <font> element defines a single font preference. Attributes: role (e.g., heading, body), family, fallback. Content model: EMPTY.

8.10. Body Section: body

The <body> element contains the primary semantic content. Attributes: usage (none|display|cache|store|train, default content usage rights). Content model: (#PCDATA | section | data | img | audio | video | link | nav)*.

8.10.1. Section: section

The <section> element groups related content. Attributes: id, label, usage (none|display|cache|store|train). Sections MAY be nested. Content model: (#PCDATA | section | data | img | audio | video | link | nav)*.

8.10.2. Image: img

The <img> element references an image resource. Attributes: src (REQUIRED, URI), inference (none|optional|required), type (media type), width, height, usage (none|display|cache|store|train). Content model: description?. A <description> child SHOULD be provided.

The inference attribute indicates whether the agent is expected to perform inference on the media resource: "none" means the resource need not be retrieved or analyzed, "optional" means the agent MAY perform inference if beneficial, and "required" means the resource contains information essential to the document that cannot be conveyed through the <description> alone.

8.10.3. Audio: audio

The <audio> element references an audio resource. Attributes: src (REQUIRED, URI), inference (none|optional|required), type (media type), duration (seconds), lang (BCP 47), usage (none|display|cache|store|train). Content model: transcript?, description?. The inference attribute has the same semantics as defined for <img>.

8.10.4. Video: video

The <video> element references a video resource. Attributes: src (REQUIRED, URI), inference (none|optional|required), type (media type), duration (seconds), width, height, lang (BCP 47), usage (none|display|cache|store|train). Content model: transcript?, description?. The inference attribute has the same semantics as defined for <img>.

8.10.5. Description: description

The <description> element provides a textual description of a media resource (<img>, <audio>, or <video>). When present, agents MAY use the description instead of performing inference on the media resource. Content model: (#PCDATA).

8.10.6. Transcript: transcript

The <transcript> element provides a textual transcript of an <audio> or <video> resource. When present, agents MAY use the transcript instead of performing inference on the media resource. Content model: (#PCDATA).

8.10.7. Link: link

The <link> element references a related resource. Attributes: href (REQUIRED, URI), rel (relationship type, e.g., "alternate", "related", "canonical"), type (media type of the referenced resource), label (human-readable description). Content model: EMPTY.

8.10.8. Data: data

The <data> element contains structured data as a collection of items. Attributes: id, label, usage (none|display|cache|store|train). Content model: item*.

8.10.9. Item: item

The <item> element represents a single record within a <data> collection. Attributes: id (string). Content model: field*.

8.10.10. Field: field

The <field> element represents a named value within an <item>. Attributes: name (string), type (string|number|boolean|date|datetime|uri). Content model: (#PCDATA).

8.10.11. Navigation: nav

The <nav> element provides pagination and continuation controls. Attributes: next (URI for the next page), prev (URI for the previous page), cursor (opaque continuation token), total (total item count, if known). Content model: EMPTY.

8.11. ANML Type System

The following type vocabulary is used by the type attribute of both the <ask> and <field> elements. This section is the single normative definition; both elements reference it. Implementations MUST apply these definitions consistently across both contexts.

string A sequence of Unicode characters. No maximum length is defined by this specification; services and agents SHOULD impose implementation-defined limits. The value attribute in <answer> and the value attribute in <field> carry string values as UTF-8 text.

number A numeric value represented as an IEEE 754 double-precision floating-point number. Implementations MUST handle precision loss gracefully. For financial values, services SHOULD use string type with explicit format documentation to avoid floating-point precision issues.

boolean One of the literal string values "true" or "false". Implementations MUST NOT accept other representations (e.g., "1", "0", "yes", "no").

date A date value in the format defined by RFC 3339 [RFC3339] full-date production: YYYY-MM-DD. Example: "2026-07-14". Implementations MUST reject values that do not conform to this format.

datetime A datetime value in the format defined by RFC 3339 [RFC3339] date-time production, in UTC with Z suffix: YYYY-MM-DDTHH:MM:SSZ. Example: "2026-07-14T09:00:00Z". Implementations MUST reject values that do not conform to this format. Local time offsets are not permitted — all datetime values MUST be UTC.

uri A URI as defined in RFC 3986 [RFC3986]. Implementations SHOULD validate that the value is a well-formed URI before transmitting it in an <answer>.

The type attribute is advisory when present on <ask> — it communicates the service's expected value format to the agent. It is descriptive when present on <field> — it communicates the data type of a value in a result set. In both cases, agents SHOULD validate or format values according to the type definition before transmission.

8.12. Footer Section: footer

The <footer> element contains copyright, licensing, and attribution declarations. Content model: rights*, attribution*, (#PCDATA)*.

8.12.1. Rights: rights

The <rights> element declares copyright and usage terms. Attributes: holder, year, license, usage (none|display|cache|store|train, default "none"), scope. Usage values form a hierarchy: none < display < cache < store < train. Multiple <rights> elements MAY target specific element IDs via the scope attribute. Content model: (#PCDATA).

8.12.2. Attribution: attribution

The <attribution> element provides credit to content sources. Attributes: required (boolean, "true"|"false", default "false"), scope (element id this attribution applies to). Content model: (#PCDATA).

8.13. Status Section: status

The <status> element communicates the outcome of a preceding operation. Content model: EMPTY. Attributes: code (REQUIRED, machine-readable status code), result (REQUIRED: success|error|partial), message (human-readable description), retry-after (seconds before the agent SHOULD retry).

9. Interaction Model

ANML operates over HTTP. It does not replace or modify HTTP methods, headers, authentication, session management, or cookie handling.

9.1. Transport

ANML interactions use HTTP [RFC9110]. The <action> element's method attribute specifies the HTTP method. ANML does not define or constrain HTTP method semantics.

9.2. Interaction Flow

A typical ANML interaction proceeds as follows:

1. Resource Retrieval: The agent issues an HTTP GET; the service returns an ANML document.

2. Document Interpretation: The agent processes the constraints, state, interact, knowledge, and body sections. If a <flow> element is present, the agent SHOULD use it to understand the full interaction sequence.

3. Disclosure Evaluation: The agent evaluates <ask> elements against user consent, local policies, trust level, sensitivity, and applicable constraints.

4. Action Selection: The agent resolves the <ask>'s action attribute to a matching <action> element to determine the method and endpoint.

5. Response Submission: The agent performs the HTTP request using the resolved method and endpoint.

6. State Advancement: The service responds with updated state, knowledge, or new interaction options.

9.3. Action Binding

All executable operations MUST be defined in the <interact> section. The agent resolves the response target by: (1) reading the action attribute of the <ask>, (2) locating the matching <action> element, and (3) using its method and endpoint to submit the response.

9.4. Knowledge Exchange

ANML defines a bidirectional knowledge exchange model. Services include <inform> and <ask> elements in documents; agents include <answer>, <refuse>, <ask>, and <inform> elements in responses.

9.5. Constraints on Disclosure

Agents MUST evaluate constraints before submitting any <answer>. Constraints may govern consent requirements, authentication, field-level disclosure restrictions, and rate limitations.

9.6. Separation of Concerns

ANML separates transport and identity (HTTP) from semantics and interaction (ANML).

10. Characters and Internationalization

The ANML internationalization model builds on XML [XML], Unicode [UNICODE], and RFC 2070 [RFC2070].

10.1. Document Character Set

The document character set is Unicode [UNICODE], synchronized with ISO/IEC 10646. Numeric character references resolve via Unicode code points regardless of encoding. ANML does not support non-Unicode character repertoires.

10.2. Character Encoding

The default encoding is UTF-8 [RFC3629]. Non-Unicode encodings MUST NOT be used. Encoding detection precedence: (1) HTTP Content-Type charset, (2) XML declaration encoding attribute, (3) BOM detection, (4) UTF-8 default.

10.3. Language Identification

The ’lang’ attribute specifies natural language using BCP 47 [BCP47] tags. Language declarations inherit from parent to child elements. ANML does not define a ’dir’ attribute; bidirectional text is handled by the Unicode Bidirectional Algorithm [UAX9].

11. Agent Response Format

This section defines the format, content model, and normative behavior for agent-originated ANML documents — documents submitted by an agent to a service in response to an <ask> or as part of a multi-step interaction.

11.1. Response Document Structure

An agent response is a conforming ANML document submitted as the body of an HTTP request to the endpoint specified by the <action> element referenced in the originating <ask>. The Content-Type of the request MUST be either ’application/anml+xml’ or ’application/anml+json’, matching the serialization used. The agent SHOULD use the same serialization as the document it is responding to unless the service has indicated a preference via the Accept header of a prior response.

The root element of an agent response document is <anml> with the same namespace and version as defined in this specification. An agent response document MAY contain any combination of the following elements in its knowledge section:

* <answer> — providing information requested by an <ask>

* <refuse> — explicitly declining to answer an <ask>

* <ask> — requesting information from the service before proceeding

* <inform> — providing unrequested contextual information the agent determines is relevant

An agent response MUST contain at least one <answer> or <refuse> element corresponding to each <ask> element marked required="true" in the originating document. An agent MAY include <answer> or <refuse> elements for optional <ask> elements at its discretion. An agent MUST NOT include <answer> elements for fields not requested by an <ask> in the current interaction context.

An agent response MUST NOT include <interact>, <persona>, <aesthetic>, <constraints>, or <state> sections. These sections are reserved for service documents. A service receiving an agent response that includes these sections MUST ignore them.

11.2. Minimum Valid Response

The minimum valid agent response is an ANML document containing a single <refuse> element for each required <ask> in the originating document. This indicates the agent has processed the document but is unable or unwilling to provide the requested information. Example:

```
<?xml version="1.0" encoding="UTF-8"?>
<anml xmlns="urn:ietf:params:xml:ns:anml:1.0">
  <knowledge>
    <refuse field="airline"
            reason="user-denied"
            message="User declined to share airline preference."/>
  </knowledge>
</anml>
```

11.3. Service Response to Agent Submission

A service receiving an agent response MUST return an HTTP response with an appropriate status code. When the service returns an ANML document as the response body, it MUST set Content-Type to the appropriate ANML media type. The service response SHOULD include a <status> element indicating the outcome of the submitted action.

The following HTTP status codes have defined semantics in the ANML interaction model:

* 200 OK — the action was processed successfully; the response body MAY contain an updated ANML document advancing the interaction state.

* 400 Bad Request — the agent response was malformed or missing required fields; the service SHOULD include a <status> element with a human-readable message.

* 401 Unauthorized — authentication is required; the agent SHOULD surface this to the user and MUST NOT retry automatically with stored credentials.

* 403 Forbidden — the action is not permitted for this agent or user; the agent MUST NOT retry the same action.

* 409 Conflict — the request conflicts with the current state of the target resource as defined by HTTP semantics. For example, attempting to book a seat that has just been reserved by another user. The service SHOULD provide updated state in the response body. Note: inventory unavailability (e.g., out of stock) is NOT a conflict in the HTTP sense and SHOULD be represented as 422 with a domain-specific status message.

* 422 Unprocessable Content — the action was understood but cannot be executed as submitted, including cases where inventory is unavailable, a field value is out of range, or a prerequisite step has not been completed. The service SHOULD indicate which fields require correction or what precondition was not met.

For 4xx responses, services SHOULD return a response body conforming to RFC 9457 [RFC9457] (Problem Details for HTTP APIs), using Content-Type 'application/problem+json' or 'application/problem+xml'. RFC 9457 supersedes RFC 7807 and is backward compatible with it. Problem Details provide machine-parseable error information including field-level validation errors, constraint violations, and retry guidance that agents can act on without human intervention. The ANML <status> element is used for 2xx responses only; 4xx and 5xx responses SHOULD use RFC 9457 Problem Details as the error envelope.

* 429 Too Many Requests — the agent is submitting too frequently; the agent MUST respect the Retry-After header if present and MUST NOT retry without it.

* 500 Internal Server Error — the service encountered an unexpected condition. The agent SHOULD NOT retry immediately. The agent MAY retry once after a delay of not less than 5 seconds. The agent SHOULD inform the user that the service encountered an error.

* 503 Service Unavailable — the service is temporarily unavailable. The agent MUST respect the Retry-After header if present. If no Retry-After header is present, the agent SHOULD NOT retry within the current session. The agent MUST inform the user that the service is unavailable.

* Other 5xx responses — the agent SHOULD treat these as equivalent to 500 unless more specific guidance is provided by the service. The agent MUST NOT interpret a 5xx response as a constraint violation or trust failure. Server errors are operational, not adversarial, unless accompanied by other indicators of malicious behavior.

11.4. Error Handling

The following normative error handling requirements apply to conforming agents and services:

* Required field absent: if a required attribute (e.g., field on <ask>, id on <action>) is absent, the agent MUST treat the containing element as malformed and MUST NOT process it. The agent SHOULD log the error and SHOULD inform the user if the malformed element would have been part of the interaction.

* Unknown step reference: if <context> references a step id that does not exist in <flow>, the agent MUST ignore the <context> element and MUST NOT attempt to infer the current workflow position.

* Action endpoint unreachable: if the HTTP request to an action endpoint fails with a network error, the agent MUST NOT retry automatically more than once. The agent SHOULD inform the user of the failure.

* Document parse failure: if an ANML document cannot be parsed as well-formed XML or valid JSON, the agent MUST NOT process any part of the document and MUST inform the user that the service returned an invalid document.

* Circular flow reference: if a <flow> contains steps that reference each other in a cycle with no exit condition, the agent MUST detect this and terminate processing of the <flow> element.

* Conflicting constraints: if two <disclosure> elements conflict for the same field, the agent MUST apply the more restrictive constraint.

11.5. Pagination Behavior

The <nav> element within <body> provides next, prev, and cursor references for multi-page result sets. The following normative behavior applies:

* Agents SHOULD follow pagination only when the user has expressed interest in additional results or when the agent determines that additional results are necessary to fulfill the user’s request.

* Agents SHOULD NOT automatically fetch all pages of a paginated result set without user awareness.

* Implementations SHOULD impose a maximum of 10 pages per interaction to prevent runaway pagination. Implementations MAY exceed this limit with explicit user authorization.

* Services MUST NOT use pagination as a mechanism to force agents to make an unbounded number of requests. A service SHOULD indicate the total number of available items in the total attribute of <nav> where known.

* Agents MUST detect and terminate pagination loops: if a next URI matches a URI already retrieved in the current interaction, the agent MUST stop following pagination.

11.6. Examples

Example agent response with multiple answers and a counter-ask:

```
<?xml version="1.0" encoding="UTF-8"?>
<anml xmlns="urn:ietf:params:xml:ns:anml:1.0">
  <knowledge>
    <answer field="destination" value="LAX" consent="explicit"/>
    <refuse field="airline"
            reason="user-denied"
            message="User prefers not to share airline preference."/>
    <ask field="available-dates"
         action="submit-booking"
         purpose="Required to show available flights"/>
    <inform>User prefers morning departures before 10:00 AM.</inform>
  </knowledge>
</anml>
```

Example service response with pagination and status:

```
<?xml version="1.0" encoding="UTF-8"?>
<anml xmlns="urn:ietf:params:xml:ns:anml:1.0">
  <status code="200" result="success"
          message="Found 47 flights matching your criteria."/>
  <body>
    <data id="flights" label="Available flights">
      <item id="f1">
        <field name="airline" type="string">Example Air</field>
        <field name="departure"
               type="datetime">2026-05-01T08:00Z</field>
        <field name="price" type="number">349</field>
      </item>
      <item id="f2">
        <field name="airline" type="string">Globe Wings</field>
        <field name="departure"
               type="datetime">2026-05-01T10:30Z</field>
        <field name="price" type="number">412</field>
      </item>
    </data>
    <nav next="/flights?page=2" total="47" cursor="eyJwYWdlIjoyfQ"/>
  </body>
  <state>
    <context><step>select</step></context>
  </state>
</anml>
```

12. Site Trust Delegation

ANML documents may be served by parties other than the site whose identity they assert. A large retailer may serve brand content under a commercial authorization. A small independent retailer may carry products without any formal authorization relationship. A CDN may serve cached documents on behalf of an origin. In all of these cases, TLS certificate verification establishes only that the agent is talking to the serving domain — it says nothing about whether that domain is authorized to assert the identity of another site.

This section defines the ANML Trust Delegation mechanism. It is modeled on the DKIM pattern: a small DNS TXT record at the site's domain serves as the single bootstrap entry point, advertising where authorization data can be found. The authorization data itself lives in a static manifest, a live query endpoint, or both — but never in DNS directly. This keeps DNS records small and stable regardless of how many partners a site authorizes, while giving sites the flexibility to choose the authorization mechanism that fits their scale and operational requirements.

12.1. Trust Tiers

ANML defines three trust tiers for third-party serving parties. The site's own canonical domain operates as an implicit Tier 3 — full trust, no verification required.

Tier 0 — Unlisted No authorization record exists for the serving domain, or no DNS bootstrap record exists for the asserted site, or verification fails for any reason. This is the default for any third-party domain. Permitted: the serving party MAY assert its own identity fully; MAY include <site-ref> elements pointing to other sites' canonical ANML documents; MAY include a <trust> element or trust-manifest attribute to invite the agent to attempt verification (the invitation itself is not an assertion). Prohibited: the serving party MUST NOT assert any identity sections (<persona>, <aesthetic>, <knowledge>, <interact>, <constraints>) on behalf of any other site. Verification failure always results in Tier 0. Agents MUST NOT apply partial trust on a failed verification.

Tier 1 — Display Authorization The serving domain is authorized with

a restricted scope. Appropriate for CDNs, aggregators, and display-only resellers. Permitted sections: "aesthetic", "body", and "knowledge" limited to confidentiality="public" inform elements. Prohibited regardless of scope array: "persona", "interact", "constraints". The tier integer is normative; scope arrays for Tier 1 entries that include prohibited sections MUST be silently truncated by the agent to exclude those sections.

Tier 2 — Full Authorization The serving domain is authorized with an explicit scope defined by the site. Appropriate for authorized retail partners and licensed distributors. The scope array defines exactly which sections may be asserted. The tier integer is normative and takes precedence over the scope array if they conflict.

12.2. DNS Bootstrap Record

The DNS bootstrap record is the single entry point for trust discovery. It is a DNS TXT record published by the site being represented at a fixed subdomain convention. Its only purpose is to advertise where authorization data can be found. It contains no authorization data itself — keeping the record small and stable regardless of how many partners a site authorizes.

The record format follows the tag-list convention established by DKIM [RFC6376] (Section 3.2) and SPF [RFC7208] (Section 4.6): a sequence of key=value pairs separated by semicolons, with defined parsing semantics. Agents MUST parse _anml TXT records using the tag-list rules defined in this section, which are consistent with those RFCs.

Rationale for reusing this format: the tag-list convention from RFC 6376 and RFC 7208 is already implemented by every major DNS management platform, monitoring tool, and DNS analysis service. Reusing the format means operators can inspect, validate, and debug _anml records using existing tooling without new software. This is a deliberate operational engineering choice, not a dependency on email authentication semantics.

The record is published at:

_anml.{site-domain} IN TXT "v=anml1; [manifest=URI;] [query=URI]"

Example:

```
_anml.example.com IN TXT "v=anml1; manifest=https://example.com/.well-known/anml-trust
; query=https://trust.example.com/anml/authorize"
```

12.2.1. Tag-List Syntax

The following ABNF [RFC5234] defines the _anml TXT record syntax. This grammar is consistent with the DKIM tag-list grammar defined in RFC 6376 Section 3.2 and the SPF record grammar in RFC 7208 Section 4.6.

```
anml-record    = tag-list
tag-list       = tag *( ";" tag ) [ ";" ]
tag            = tag-name "=" tag-value
tag-name       = ALPHA *ALNUMPUNC
tag-value      = *( %x21-3A / %x3C-7E )
                 ; printable ASCII except ";" (%x3B)
                 ; "=" (%x3D) is permitted in tag-value
                 ; (e.g., within URI query strings)
ALNUMPUNC      = ALPHA / DIGIT / "_" / "-" / "."
version-tag    = %s"v" "=" %s"anml1"
manifest-tag   = %s"manifest" "=" URI
query-tag      = %s"query" "=" URI
URI            = <URI as defined in RFC 3986, Section 3>
```

Parsing rules (normative):

* Tag names are case-sensitive. "v=anml1" is valid; "V=anml1" MUST be ignored as unrecognized.

* Whitespace (SP, HTAB) before or after "=" or ";" MUST be ignored by parsers. Implementations MUST NOT include whitespace in generated records, but MUST tolerate it when parsing.

* Tag pairs MUST be split on the first ";" delimiter only. Within each tag pair, the tag-name and tag-value MUST be split on the FIRST "=" character only. All subsequent "=" characters within the tag-value are part of the value — for example, a URI query string such as "https://example.com/auth?a=1&b=2" is a single tag-value. Parsers MUST NOT split on "=" characters after the first "=" in a tag pair.

* Duplicate tag names within a single record are prohibited. If duplicate tag names are present, the agent MUST treat the entire record as malformed and ignore it.

* Unknown tag names MUST be ignored without error. This allows forward compatibility with future tag definitions.

* DNS TXT records are limited to 255 octets per string. Multiple strings in a single TXT RRset are concatenated in order per RFC 1035 Section 3.3.14. Agents MUST concatenate all strings in an _anml TXT record before parsing. If a URI value is split across string boundaries, it MUST be reassembled before use.

* A record without a recognized "v" tag MUST be ignored entirely. A record where v != "anml1" MUST be ignored. Agents MUST process a record only after confirming v=anml1. The "v" tag MUST appear first.

* At least one of "manifest" or "query" MUST be present in a valid record. A record with neither is malformed and MUST be ignored.

12.2.2. Record Fields

v Version tag. REQUIRED. MUST be the first tag in the record per DKIM convention (RFC 6376 Section 3.5). Current value: "anml1".

manifest URI of the static Trust Manifest JSON document. MUST use the https scheme. MUST be served from the site’s canonical domain. OPTIONAL if query is present.

query URI of the live Trust Query endpoint. MUST use the https scheme. MAY be served from any domain — its authority derives from publication in the site’s own DNS record. When DNSSEC is not available for the _anml record, agents SHOULD prefer the manifest (which requires TLS cert verification against the site domain) over the query endpoint. When DNSSEC is verified, agents MAY use the query endpoint without additional domain verification. OPTIONAL if manifest is present.

12.2.3. Absent or Invalid Records

If no _anml TXT record exists for the asserted site domain, the agent MUST treat all third-party serving parties as Tier 0 for that site. The absence of a DNS record means the site has not published trust delegation — not that the site is absent or does not support ANML.

A site domain MAY publish multiple _anml TXT records. Agents MUST process all records with a recognized version tag. If multiple records advertise both manifest and query URIs, agents SHOULD prefer the query endpoint.

Sites SHOULD enable DNSSEC for their _anml records. Agents SHOULD verify DNSSEC signatures where available. A DNSSEC validation failure MUST be treated as a verification error — the agent MUST fall back to Tier 0 and MUST NOT treat the failure as equivalent to a missing record.

12.3. Static Trust Manifest

The static Trust Manifest is a JSON document served at the URI advertised in the _anml DNS record. It is appropriate for sites with a small, stable set of authorized partners requiring no additional infrastructure beyond a web server. Served with Content-Type 'application/anml-trust+json' over HTTPS from the site's canonical domain.

```
{
  "site": "example.com",
  "version": "1.0",
  "issued": "2026-05-01T00:00:00Z",
  "expires": "2026-08-01T00:00:00Z",
  "authorized-domains": [
    {
      "domain": "*.cdn.example.com",
      "tier": 1,
      "scope": ["aesthetic", "body"],
      "note": "CDN — display only"
    },
    {
      "domain": "example.net",
      "tier": 2,
      "scope": ["aesthetic", "persona", "knowledge"],
      "note": "Authorized retailer"
    },
    {
      "domain": "example.org",
      "tier": 2,
      "scope": ["aesthetic", "persona", "knowledge", "interact"],
      "note": "Preferred retailer — full delegation"
    }
  ]
}
```

Fields:

site Canonical domain of the site. REQUIRED. Agents MUST verify this matches the domain in the _anml DNS record used to discover this manifest. If it does not match, the manifest MUST be rejected and the agent MUST treat the serving party as Tier 0.

version Manifest format version. REQUIRED. Current: "1.0".

issued ISO 8601 datetime of issuance. REQUIRED.

expires ISO 8601 datetime in UTC (Z suffix REQUIRED) at which this manifest expires. REQUIRED. RECOMMENDED maximum: 90 days from issued. Agents MUST NOT use an expired manifest and MUST treat an expired manifest as equivalent to a missing manifest — Tier 0.

authorized-domains Array of authorization records. REQUIRED. An empty array means no third-party domains are authorized — all serving parties operate at Tier 0 for this site.

Each authorization record:

domain Authorized domain or wildcard. A wildcard of the form '*.example.com' matches any single DNS label prepended to 'example.com' per RFC 1034, and does NOT match the apex domain itself or any multi-level subdomain. REQUIRED.

tier Integer 1 or 2. Normative. Takes precedence over scope when they conflict. REQUIRED.

scope Array of ANML section names the domain is authorized to assert. Valid values: "head", "persona", "aesthetic", "knowledge", "interact", "constraints", "body", "footer". REQUIRED.

note Human-readable description. OPTIONAL.

12.4. Live Trust Query Endpoint

The live Trust Query endpoint allows a site to answer per-domain authorization queries in real time without publishing a manifest listing every authorized partner. Appropriate for sites with large or frequently changing partner sets, or sites requiring real-time revocation. The endpoint URI is authoritative because it is published in the site's own _anml DNS record. When DNSSEC is verified for the _anml record, agents MAY use the query endpoint without additional TLS domain verification. When DNSSEC is unavailable, agents SHOULD prefer the manifest, which requires TLS certificate verification against the site domain.

Scope of this mechanism: the trust delegation defined in this section applies exclusively to ANML document serving parties. It does not apply to Trust Manifest or Trust Query endpoint responses, which are processed as raw JSON data. Agents MUST NOT apply ANML document parsing, trust tier evaluation, or <trust> element processing to responses received from Trust Manifest or Trust Query endpoints — these responses are authorization data, not ANML documents.

Query:

```
GET {query-uri}?domain={serving-domain}&scope={comma-separated-sections}
Accept: application/json
```

Example:

```
GET https://trust.example.com/anml/authorize?domain=example.net&scope=persona,aesthetic,k
nowledge
```

Response when authorized (HTTP 200):

```
{
  "authorized": true,
  "tier": 2,
  "scope": ["persona", "aesthetic", "knowledge"],
  "expires": "2026-08-01T00:00:00Z"
}
```

Response when not authorized: the endpoint MUST return HTTP 404 (Not Found) with an empty body or minimal diagnostic JSON. The endpoint MUST NOT return HTTP 200 with an "authorized": false body. Returning 404 for unauthorized domains prevents enumeration of a site’s authorized partner list: an agent or third party probing the endpoint cannot distinguish "not authorized" from "we have no record of this domain," which limits competitive intelligence leakage. Agents MUST treat a 404 response as a definitive negative authorization and MUST cache it as such for the duration of the applicable DNS TTL of the _anml record.

Response fields (HTTP 200 only):

authorized Boolean true. REQUIRED. Always true in a 200 response — a negative result is indicated by HTTP 404, not by this field.

tier Integer 1 or 2. REQUIRED. Normative — takes precedence over the scope array if they conflict.

scope Array of authorized section names. REQUIRED. MAY be a subset of the requested scope. The agent MUST use only the returned scope, not the requested scope.

expires ISO 8601 datetime in UTC (Z suffix REQUIRED) after which this authorization MUST be revalidated. REQUIRED. Agents MUST cache 200 responses until this time. Agents SHOULD revalidate in the background before expiry to avoid authorization gaps.

HTTP status codes:

* 200 OK — domain is authorized. Body contains tier, scope, and expires.

* 404 Not Found — domain is not authorized. Agent MUST cache this as a negative result.

* 400 Bad Request — malformed query. Agent MUST treat as Tier 0 and SHOULD log.

* 429 Too Many Requests — agent MUST respect Retry-After and MUST treat as Tier 0 for the current interaction.

* 5xx Server Error — transient. Agent SHOULD fall back to the static manifest if available in the DNS record. If unavailable, agent MUST treat as Tier 0.

Cache key: agents MUST cache query results keyed on the tuple (serving-domain, site-domain). The cached value is the full authorization record including tier, scope, and expires. When a new request arrives, the agent checks the cache for (serving-domain, site-domain), retrieves the full authorization, and applies the intersection of the cached scope with the requested scope locally — without issuing a new network request. The endpoint is queried at most once per (serving-domain, site-domain) pair per TTL period regardless of the scope requested. The same cache key applies to both manifest lookups and query endpoint responses.

12.5. The trust Element and Verification Procedure

An ANML document asserting a third-party site identity MUST include either a <trust> element in its <head> section (single-site documents) or a the <site> element with a domain attribute differing from the serving domain (multi-site documents). Without one of these, the agent MUST treat all third-party site identity assertions as Tier 0.

In single-site documents the <trust> element carries only the site domain — the agent discovers the manifest and query endpoints via DNS, keeping the ANML document itself free of authorization infrastructure details:

```
<!-- Single-site -->
<head>
  <title>Example Product 30 oz — Independent Retailer</title>
  <trust domain="example.com"/>
</head>

<!-- Multi-site -->
<site domain="example.com">
  <persona>...</persona>
</site>
```

Attributes of <trust>:

domain Canonical domain of the site being asserted. REQUIRED. The agent uses this value to look up the _anml DNS TXT record.

Agent verification procedure:

1. Extract site domain from <trust>/@site or <site>/@domain.

2. Look up _anml.{site-domain} TXT records via DNS. If no record exists with a recognized version tag, treat as Tier 0.

3. Parse the record for manifest and query URIs.

4. Check cache for a valid non-expired authorization for the tuple (serving-domain, site-domain). If found, use it and skip to step 9.

5. If a query URI is advertised, send a query request per Section 12.4. On 200 response, proceed to step 8. On 5xx or network error, fall back to step 6.

6. If a manifest URI is advertised (and query was unavailable or failed), fetch the manifest over HTTPS. Verify TLS certificate matches the site domain. If fetch fails or certificate is invalid, treat as Tier 0. Verify the manifest site field matches the asserted site domain. Verify the manifest has not expired.

7. Search authorized-domains for the serving domain, using wildcard matching per RFC 1034. If not found, treat as Tier 0.

8. Extract tier and scope from the query response or manifest entry.

9. Apply Tier 1 or Tier 2 permissions per Section 12.1. Process only in-scope sections as site assertions. Cache the result with its expiry. Sections not in scope are treated as assertions of the serving domain only, subject to Tier 0 rules.

Any verification failure at any step MUST result in Tier 0 for the asserted site. Agents MUST NOT apply partial trust. The serving domain's own identity is unaffected by a failed verification for a third-party site.

12.6. The site-ref Element

A Tier 0 serving party that carries content from third-party sites MAY use <site-ref> to point agents to those sites' canonical ANML documents. This is a pointer, not a delegation — the serving party makes no identity assertions about the referenced site.

```
<head>
  <title>Independent Retailer — Outdoor Gear</title>
  <site-ref domain="example.com"
            canonical="https://example.com/.well-known/anml"
            relationship="carries-products"/>
</head>
```

Attributes:

domain Canonical domain of the referenced site. REQUIRED.

canonical URI of the site's own ANML document. Agents SHOULD fetch this URI for all <site-ref> elements present in a document when processing that document, up to an implementation-defined maximum number of site-ref fetches per document (RECOMMENDED maximum: 5). This is a mechanical rule — agents MUST NOT condition the fetch on an interpretation of user intent. If the fetch fails or the TLS certificate is invalid, the agent MUST treat the referenced site as unknown and MUST NOT use any cached or inferred identity for it. REQUIRED.

relationship Nature of the relationship. Values from the ANML Site Relationship Values registry (Section 15.6). OPTIONAL.

A document fetched via <site-ref> is processed as a Tier 3 document — full trust from the canonical domain. <site-ref> does not elevate the serving party's trust tier. The serving party's own identity sections remain subject to Tier 0 restrictions.

12.7. Multi-Site Documents and Layered Identity Attack Prevention

A serving party MAY publish an ANML document that presents identity for both itself and authorized third-party sites. This is a legitimate and common use case: a retailer's product page carries the retailer's own identity (interactions, service persona) alongside the product manufacturer's identity (product voice, aesthetic, knowledge). The <site> wrapper element (Section 8.2) provides the structural mechanism for this.

A document with identity assertions for more than one domain MUST use the multi-site content model: all identity sections are wrapped in <site> elements, one per domain. This replaces the single-site content model where identity sections appear directly under <anml>. The structural separation of identity by domain prevents the layered identity attack described below.

The layered identity attack: a malicious serving party blurs the boundary between its own assertions and those of a more trusted site, causing the agent to attribute the serving party's content to the trusted site, or to apply the trusted site's permission tier to the serving party's own sections. The <site> wrapper prevents this by making domain attribution structural rather than attributional — there is no attribute to forge, only a wrapper element whose domain is verified independently.

Normative rules for multi-site documents:

* Each <site> element's domain attribute MUST match exactly one domain. The agent MUST apply trust tier verification independently to each <site> element whose domain differs from the serving domain.

* A Tier 2 authorization for one site MUST NOT elevate the trust tier of the serving domain's own <site> element.

* The agent MUST NOT merge or blend identity sections from different <site> elements. Each site's persona, aesthetic, and knowledge are evaluated and applied independently.

* A <site> element containing an <interact> section is attributed exclusively to that site's domain. The agent MUST NOT execute an action from a third-party <site> element unless the trust manifest for that site's domain explicitly includes "interact" in scope.

* If a <persona> or <instructions> element within any <site> references or attempts to invoke another site's identity, the agent MUST treat this as a prompt injection attempt.

* An agent MUST be able to explain to a user, on request, which identity assertions came from which site domain and at which trust tier.

Example of a conforming multi-site document (example.net serving a Example Brand product page; example.net holds Tier 2 authorization from example.com for persona, aesthetic, and knowledge):

```
<anml xmlns="urn:ietf:params:xml:ns:anml:1.0">

  <!-- example.net's own identity — serving domain, Tier 3, no verification -->
  <site domain="example.net">
    <head>
      <title>Example Product 30 oz — Example Retailer</title>
      <site-ref site="example.com"
                canonical="https://example.com/.well-known/anml"
                relationship="carries-products"/>
    </head>
    <persona>
      <tone value="helpful"/>
      <instructions>Assist the user with purchase and delivery
      questions for this order.</instructions>
    </persona>
    <interact>
      <action id="add-to-cart" method="POST"
              endpoint="/cart/add" confirm="true"/>
    </interact>
  </site>

  <!-- Example Brand's authorized identity — Tier 2, scope verified by agent -->
  <!-- Agent automatically looks up _anml.example.com for verification -->
  <site domain="example.com">
    <persona>
      <tone value="rugged"/>
      <instructions>Speak to Example Brand's product quality, materials,
      and warranty. Do not discuss pricing or delivery.</instructions>
    </persona>
    <aesthetic>
      <display-name>Example Brand</display-name>
    </aesthetic>
    <knowledge>
      <inform>The Rambler 30 oz is made from 18/8 stainless steel
      and is dishwasher safe.</inform>
    </knowledge>
  </site>

</anml>
```

An agent processing this document MUST apply Example Brand's persona and knowledge only in the context of product information, apply example.net's persona for purchase and logistics, and attribute the add-to-cart action exclusively to example.net. The agent MUST NOT represent the add-to-cart action as an example.com-defined interaction because "interact" is not in Example Brand's authorized scope for example.net.

Note: the <trust> element used in single-site trust verification (Section 12.3) is replaced in multi-site documents by the domain and trust-manifest attributes on the <site> element. Both mechanisms initiate the same agent verification procedure defined in Section 12.4.

12.8. Security Considerations for Trust Delegation

The Trust Delegation mechanism is bounded by the security of the brand's canonical domain. A compromised brand domain can issue fraudulent Trust Manifests. Brands SHOULD:

* Set manifest expiry to 30-90 days maximum.
* Scope authorizations as narrowly as commercially necessary.
* Audit authorized-domains quarterly and remove inactive partners immediately.
* Use DNSSEC to prevent DNS-based manifest hijacking.
* Monitor for unauthorized <trust> elements claiming their domain from third-party sites.

Agents MUST NOT:

* Accept Trust Manifests over unencrypted HTTP.
* Accept self-signed certificates for manifest endpoints.
* Cache manifests beyond their expires value.
* Apply a Tier 2 scope to a domain listed as Tier 1.
* Allow <site-ref> to elevate the serving party's trust tier.

13. Security Considerations

ANML defines document structure and interaction semantics. It does not enforce security, privacy, or access control at the protocol level. Agent developers and service operators are responsible for implementing safety, privacy, and policy controls appropriate to their deployment context. This section provides a structured threat model and MUST-level guidance for conforming implementations.

ANML documents are processed by autonomous agents that may act without direct human oversight at each step. The attack surface includes the document itself, the service publishing it, the agent processing it, and the transport layer.

13.1. Threat Model

The following threat actors and scenarios are in scope for this specification:

* A malicious service publishing a crafted ANML document to manipulate agent behavior, extract user data, or cause the agent to execute harmful actions.

* A malicious agent submitting false or manipulated answers to exploit service logic, obtain resources it is not authorized to receive, or poison service-side state.

* A network adversary intercepting or modifying ANML documents in transit to alter interaction semantics or inject malicious content.

* A malicious document delivered via a compromised or typosquatted well-known URI, targeting agents that discover ANML documents automatically.

* An honest but misconfigured service that inadvertently exposes sensitive information through poorly specified ask elements or unconstrained inform elements.

The following are explicitly out of scope: compromise of the agent runtime environment, attacks on the user’s device or operating system, and attacks that exploit vulnerabilities in the underlying XML parser beyond those addressed in this section.

13.2. Information Disclosure via Knowledge Exchange

The <ask> element requests information from the agent on behalf of a service. This presents a significant data exfiltration risk if agents respond without rigorous evaluation of consent and constraints.

Conforming agents MUST:

* Evaluate all applicable <constraints> before responding to any <ask>.

* Obtain explicit or implicit user consent, as required by the applicable <disclosure> element, before disclosing any information.

* Treat the absence of a <constraints> section as equivalent to requires="none" for all fields — not as authorization to disclose without evaluation.

* Apply the principle of minimum necessary disclosure: provide only the fields explicitly requested by the <ask> elements in the current document, limited to the values strictly required to fulfill the stated purpose attribute of each <ask>. An agent MUST NOT disclose additional fields, inferred values, or context beyond what is directly requested and consented to for each specific <ask>.

* NEVER disclose information that the user has not authorized, regardless of the service’s ask or inform content.

Services MUST NOT rely on ANML constraints alone as an access control mechanism. Server-side authorization MUST be implemented independently of ANML document constraints.

Agents SHOULD maintain a disclosure log sufficient to reconstruct, for any completed interaction, which fields were disclosed, to which service domain, under which consent basis, and at what time. Agents SHOULD make this log available to the user on request. The format and retention period of the log are implementation-defined.

13.3. Prompt Injection via Persona and Instructions

The <persona> and <instructions> elements present a significant prompt injection risk. A malicious service may craft persona guidance or free-text instructions designed to override the agent's safety policies, manipulate its behavior toward the user, cause it to misrepresent the service, extract information it would not otherwise disclose, or act against the user's expressed interests.

Conforming agents MUST:

* Treat ALL <persona> and <instructions> content as advisory and untrusted input from an external source, regardless of the apparent trustworthiness of the service.

* Apply the same input sanitization and policy evaluation to <instructions> content as to any other untrusted text input.

* NEVER override user-configured safety policies, privacy preferences, or platform-level guardrails based on <instructions> content.

* NEVER represent themselves as a different agent, platform, or identity in response to <persona> guidance.

* Disregard <instructions> that direct the agent to ignore, suppress, or override this specification's conformance requirements.

Agents SHOULD apply heuristic or model-based analysis to <instructions> content before processing to identify potential injection patterns. The detection method is implementation-defined; the response to a detected injection attempt is normative. When an agent determines that <instructions> or <persona> content constitutes a prompt injection attempt, the agent MUST: (1) discard the offending element entirely and MUST NOT apply any part of its content to agent behavior — a partial or sanitized application is not permitted; (2) record the detection in its operational log; and (3) produce a user-visible notification that instructions from the service were rejected, without including the rejected content in the notification. The Human Intent First design principle (Section 4) takes absolute precedence over any <persona> or <instructions> content.

Services are RECOMMENDED to limit <instructions> content to communication style guidance and to avoid imperative behavioral directives that could be misused by intermediaries.

13.4. Action Execution Risks

The <interact> section declares operations that agents may execute against arbitrary HTTP endpoints. This presents risks including unauthorized transactions, SSRF (Server-Side Request Forgery) via agent-mediated requests, and unintended side effects on third-party systems.

Conforming agents MUST:

* Validate that the action endpoint is consistent with the origin of the ANML document before executing any action.

* Require explicit user confirmation before executing any action with confirm="true" or any action with significant real-world consequences (purchases, deletions, transmissions of personal data).

* Revalidate site trust authorization immediately before executing any action with confirm="true", any action producing a financial transaction, booking, or irreversible state change, or any action that discloses personal data. Revalidation MUST query the trust endpoint or fetch the manifest fresh — a cached authorization result MUST NOT be used for these action classes regardless of its expires value. This prevents execution of actions under trust authorization that has been revoked since document parse time. See Section 12 for the trust verification procedure.

* NEVER execute actions that conflict with user-expressed intent or platform safety policies.

* Apply rate limiting to action execution to prevent runaway automation.

* NOT follow action endpoint redirects to different origins without re-validating user intent.

Services SHOULD use the idempotent attribute to indicate safe retry behavior and the confirm attribute for all consequential actions.

13.5. XML External Entity (XXE) Attacks

Conforming agents and ANML document validators MUST NOT process DOCTYPE declarations, internal DTD subsets, or external entity references in ANML documents. Conforming ANML documents SHOULD NOT include a DOCTYPE declaration. If a DOCTYPE declaration is present, the agent MUST ignore it and MUST NOT resolve any entities defined within it. Parsers configured to resolve external entities MUST NOT

be used to parse ANML documents.

13.6. Agent Spoofing and Service Impersonation

ANML does not define agent authentication mechanisms. A service cannot cryptographically verify the identity of the agent submitting an answer, nor can an agent cryptographically verify that an ANML document originates from the claimed service without additional mechanisms.

Services that require verified agent identity SHOULD use existing HTTP authentication mechanisms (OAuth 2.0, API keys, mutual TLS) in conjunction with ANML interactions.

Agents SHOULD verify the TLS certificate of the service before processing an ANML document retrieved over HTTPS. Agents SHOULD treat ANML documents retrieved over unencrypted HTTP as untrusted and MUST NOT submit <answer> responses to services communicating over unencrypted HTTP.

Agents SHOULD apply heuristics to detect service impersonation, including verification that the well-known URI origin matches the domain of links and action endpoints in the document.

13.7. Malicious ANML Documents

A malicious ANML document may attempt to exhaust agent resources through excessive nesting depth, very large element counts, unbounded <body> content, or circular references in <flow> elements.

Conforming agents MUST impose implementation-defined limits on:

* Document size (RECOMMENDED maximum: 1MB)
* Element nesting depth (RECOMMENDED maximum: 32 levels)
* Number of <action> elements (RECOMMENDED maximum: 64 per document)
* Number of <ask> elements (RECOMMENDED maximum: 32 per document)
* Number of HTTP requests generated from processing a single document (RECOMMENDED maximum: 8)
* Total processing time per document

Agents MUST detect and terminate processing of circular <flow> references.

13.8. Constraint Bypass

The <constraints> section declares disclosure rules that conforming agents must evaluate. However, a non-conforming or malicious agent may ignore these constraints entirely. Services MUST NOT rely on ANML constraint declarations as the sole mechanism for protecting sensitive information. Server-side access control, authentication, and authorization MUST be implemented independently.

The absence of a <constraints> section does not imply authorization to disclose any particular information. Agents MUST apply default-deny disclosure policies in the absence of explicit user consent.

13.9. Replay and State Manipulation

An agent or network adversary may attempt to replay previously captured ANML responses or manipulate <state> content to advance or regress a multi-step workflow without authorization.

Services MUST NOT trust agent-provided state values without independent server-side validation. Services SHOULD use cryptographic nonces or session tokens to bind state to a specific interaction sequence. Services MAY apply HTTP Message Signatures [RFC9421] to ANML responses to detect tampering.

13.10. Denial of Service

Conforming agents SHOULD impose resource limits on ANML document processing as described in Section 11.7. Services SHOULD apply rate limiting to ANML document endpoints, particularly well-known URIs which are publicly discoverable and may be targeted by automated crawlers. Services SHOULD return HTTP 429 (Too Many Requests) with appropriate Retry-After headers when rate limits are exceeded.

13.11. Cross-Origin Considerations

Agents SHOULD apply origin-based security policies when retrieving resources referenced within ANML documents, consistent with the same-origin policy principles established for web browsers. Action endpoints that differ in origin from the ANML document SHOULD require additional user confirmation. Agents SHOULD NOT automatically follow links or retrieve resources from origins not established in the initial ANML document without user awareness.

13.12. Transport Security

Services MUST serve ANML documents over HTTPS with valid TLS certificates. Agents MUST verify TLS certificates when retrieving ANML documents. Agents MUST NOT submit <answer> elements containing personal information to services communicating over unencrypted HTTP. Agents SHOULD refuse to process <interact> action endpoints that specify HTTP (non-TLS) URIs unless the user has explicitly acknowledged the risk.

14. Privacy Considerations

ANML introduces a structured mechanism for information exchange between services and agents acting on behalf of users. This section addresses the privacy implications of that mechanism, with particular attention to the cross-session, cross-platform context portability enabled by the knowledge exchange framework.

14.1. User Control and Consent

The foundational privacy principle of ANML is that the user controls what the agent discloses. The <ask> / <answer> framework is designed as a consent mechanism, not a data extraction mechanism. The following conformance requirements govern agent disclosure behavior:

* Agents MUST NOT disclose information in response to an <ask> without first evaluating applicable constraints and determining that disclosure is permitted. This requirement is testable: a conforming agent presented with an <ask> and a <disclosure requires="explicit-consent"> constraint MUST NOT transmit the requested field value in an <answer> without that consent having been obtained. The mechanism by which consent is obtained is implementation-defined.

* Agents MUST maintain a record of disclosure decisions sufficient to produce a disclosure log on request. The format and retention period of this record are implementation-defined. A conforming agent MUST be able to produce, upon request from the user or from a conformance test, a list of fields disclosed in a given interaction and the consent basis for each.

* Agents MUST support user-directed refusal: when a user instructs an agent to refuse all <ask> requests from a specific domain, the agent MUST respond to subsequent <ask> elements from that domain with <refuse reason="user-denied"> regardless of constraint configuration. This requirement is testable.

* Agents MUST NOT disclose context received from one service domain to a different service domain without explicit per-disclosure user authorization. This is testable: a conforming agent MUST NOT include in an <answer> to service B information that was received only via <inform> from service A.

NOTE: requirements for specific user interface elements, consent dialog designs, or logging user interface features are outside the scope of this specification and are left to implementation. The conformance requirements above are expressed in terms of agent behavior, not interface design.

14.2. Cross-Session and Cross-Platform Context Portability

ANML's <inform> and <ask> / <answer> framework enables agents to carry context established in one session or with one service into subsequent sessions with other services. This capability is intentionally designed to benefit users — allowing preferences, history, and relationship context to persist across platforms without centralized data storage.

However, this same capability presents privacy risks if implemented without appropriate user controls:

* A service may attempt to obtain information about a user's interactions with other services by crafting <ask> elements that target context established elsewhere.

* An agent carrying context from a sensitive interaction (e.g., a healthcare provider) may inadvertently disclose that context to an unrelated service.

* Context carried by an agent may persist beyond the user's expectation or intent.

Conforming agents MUST:

* Not disclose to service B context received exclusively from service A, without explicit authorization scoped to that cross-service disclosure. This is testable: a conforming agent receiving <inform> from service A MUST NOT include that information in an <answer> to service B unless cross-service disclosure was explicitly authorized.

* Not disclose content marked confidentiality="private" to any service other than the one from which it was received. This is testable: confidentiality="private" content MUST NOT appear in any <answer> or <inform> element sent to a different domain.

* Not disclose sensitive categories of information (health data, financial data, location history, as identified by the purpose attribute of the originating <ask>) to a service other than the one that requested it, without per-disclosure user authorization. This is testable by the presence of sensitive-category purpose values in disclosed answers to third-party services.

* Implement context deletion on user instruction: when a user instructs an agent to delete context associated with a specific service domain, the agent MUST NOT include that context in any subsequent <answer> or <inform> to any service. Implementation of the deletion instruction interface is implementation-defined; the behavioral outcome is normative.

14.3. Data Minimization

Services SHOULD request only the information necessary for the stated purpose of the interaction. The purpose attribute of the <ask> element is REQUIRED to be present and accurate. Agents SHOULD evaluate the stated purpose and decline to disclose information where the stated purpose is implausible, vague, or inconsistent with the nature of the service.

14.4. Usage Rights and Content Controls

The usage attribute of <inform> and <rights> elements specifies what agents may do with content. Conforming agents MUST respect usage declarations. In particular:

* Content marked usage="none" MUST NOT be retained, cached, stored, or used for any purpose beyond immediate display in the current interaction.

* Content marked usage="display" MUST NOT be cached or stored beyond the current session.

* Content marked usage="cache" MAY be retained for the duration specified by the ttl attribute but MUST be discarded thereafter.

* Content marked usage="store" MAY be retained indefinitely subject to applicable data protection regulations.

* Content marked usage="train" MAY be used for model training, subject to the user’s separate consent for their agent platform’s training data policies.

Usage declarations are service assertions and are not cryptographically enforced. Agents are responsible for honoring them. Services should not rely on usage declarations as the sole mechanism for protecting content.

14.5. Regulatory Considerations

ANML implementations may be subject to data protection regulations including the EU General Data Protection Regulation (GDPR), the California Consumer Privacy Act (CCPA), the Health Insurance Portability and Accountability Act (HIPAA), and equivalent regulations in other jurisdictions. This specification does not constitute legal advice. Implementers are responsible for assessing and ensuring compliance with applicable regulations in their jurisdiction and use case.

The consent framework provided by <constraints> and <disclosure> elements is designed to be compatible with GDPR explicit consent requirements for sensitive data categories. However, ANML consent mechanisms are not a substitute for proper legal consent mechanisms required by applicable law. Services operating in regulated contexts (healthcare, finance, government) MUST implement additional consent and data protection mechanisms beyond those specified in this document.

15. Relationship to Related Standards and Formats

ANML addresses a problem space that intersects with several existing standards and formats. This section clarifies how ANML relates to each and why a new specification is warranted rather than an extension of existing work.

15.1. HTML

HTML is a presentation language for human consumption. ANML is a semantic document format for machine consumption. HTML encodes intent, interactions, and constraints implicitly through visual structure, JavaScript behavior, and user interface conventions that are inaccessible to agents without significant inference overhead. ANML makes these explicit. ANML is designed to complement HTML, not replace it: a service may serve both an HTML interface for human users and an ANML document for agent consumers at the same URL via content negotiation.

15.2. JSON-LD and Schema.org

JSON-LD is a method for encoding linked data in JSON, primarily used to provide structured data for search engine indexing and knowledge graph construction. Schema.org defines a vocabulary of types and properties for describing entities on the web.

ANML differs from JSON-LD and Schema.org in three fundamental respects:

* Interaction model: JSON-LD and Schema.org describe what things are. ANML describes how to interact with a service — defining executable actions, multi-step workflows, and bidirectional knowledge exchange. Schema.org has no equivalent of <interact>, <ask>, <answer>, or <state>.

* Consent framework: ANML defines a structured consent and disclosure model (<constraints>, <disclosure>, <refuse>) with no equivalent in JSON-LD or Schema.org.

* Agent behavioral guidance: ANML's <persona> section provides advisory guidance for agent communication style, brand voice, and vocabulary — a concern entirely outside the scope of linked data formats.

ANML documents MAY reference Schema.org types within <body> content where appropriate. The two standards are complementary rather than competing.

15.3. OpenAPI and AsyncAPI

OpenAPI (formerly Swagger) and AsyncAPI define machine-readable interface descriptions for REST and event-driven APIs respectively. These formats describe how a developer should build a client that integrates with a specific service.

ANML differs fundamentally in its discovery and integration model: an OpenAPI document describes a bespoke service contract that requires prior developer integration. ANML provides a universal document format that any conforming agent can interpret without prior service-specific programming — analogous to the difference between a custom proprietary document format and HTML. A browser does not need to be pre-programmed for each website; a conforming ANML agent does not need to be pre-programmed for each service.

ANML action definitions are intentionally simpler than OpenAPI operation objects. ANML is designed for agent-mediated user interactions, not developer API integration.

15.4. robots.txt and llms.txt

robots.txt provides access control directives for web crawlers. llms.txt (an emerging informal convention) provides hints for LLM content consumption. Neither defines structured interaction semantics, knowledge exchange, consent frameworks, or agent behavioral guidance. ANML is a richer, structured interaction layer that subsumes the agent-guidance use case of llms.txt while providing substantially greater capability.

15.5. Rationale for XML Serialization

ANML uses XML 1.0 as its primary serialization format. This choice reflects several design considerations that distinguish ANML from API formats that use JSON:

* Human authoring: ANML documents are content artifacts designed to be authored and reviewed by humans — brand managers, content strategists, healthcare administrators — in addition to being parsed by machines. The tag-based markup structure of XML is self-documenting in a way that JSON object notation is not. A non-technical author can read and understand an ANML document without developer assistance.

* Namespace extensibility: XML Namespaces provide a well-established, IETF-precedented mechanism for extending ANML documents with domain-specific vocabularies without conflicting with the core specification. This is essential for the vertical-specific profiles anticipated for healthcare, commerce, and government use cases.

* Document model: XML's document model, including processing order, whitespace handling, and character reference support, is appropriate for a format that carries rich text content in <body>, <instructions>, and <inform> elements.

* IETF precedent: XML-based formats have substantial precedent in IETF standards, including XMPP (RFC 6120), Atom (RFC 4287), and XML-based MIME type registrations.

The analogy is instructive: HTML, which ANML complements, is also an XML-serializable markup language. The choice of XML places ANML in the tradition of web content formats rather than API data formats — which accurately reflects its purpose.

Future versions of this specification may define a JSON serialization (application/anml+json) for deployment contexts where XML parsing is impractical. Any such serialization would be required to be semantically equivalent to the XML serialization defined herein.

15.6. Known Implementations

The following implementations of this specification are known to the authors at the time of publication:

* Reference server implementations in Node.js/TypeScript, Python, Go, PHP, and Rust, maintained by the ANML Foundation at github.com/anmlfoundation/implementations.

* Reference client (agent-side) implementations in Node.js/ TypeScript, Python, Go, PHP, and Rust, maintained by the ANML Foundation.

* Savo, the chat agent component of Life Savor AI (Ellevan), which implements ANML document discovery, parsing, and knowledge exchange as a production consumer agent.

The ANML Foundation actively solicits additional independent implementations and maintains a registry of known implementations at anmlfoundation.org/implementations. Organizations implementing this specification are encouraged to notify the Foundation at implementations@anmlfoundation.org to be included in the registry.

16. IANA Considerations

16.1. Media Type Registrations

This document requests registration of three media types with IANA per [RFC6838]:

* ’application/anml+xml’ — as specified in Section 6.1 of this document.

* ’application/anml+json’ — as specified in Section 7.1 of this document.

* ’application/anml-trust+json’ — as specified below.

Registration for application/anml-trust+json:

Media Type name: application

Media subtype name: anml-trust+json

Required parameters: none

Optional parameters: none

Encoding considerations: UTF-8 as specified in RFC 8259. A BOM MUST NOT be prepended.

Security considerations: See Section 13. Trust Manifest documents MUST be served over HTTPS. Agents MUST verify the TLS certificate of the serving endpoint against the site domain specified in the associated _anml DNS record. An expired Trust Manifest MUST be treated as equivalent to no manifest.

Interoperability considerations: Conforming agents MUST safely ignore unknown keys. The site field MUST match the domain of the _anml DNS record used to discover this manifest.

Published specification: This document, Section 12.3.

Applications that use this media type: ANML agents performing site trust delegation verification.

File extension(s): none defined

16.2. Well-Known URI Registrations

This document requests registration of the following Well-Known URIs per [RFC8615]:

URI suffix: anml

Change controller: IETF

Specification document: This document, Section 6.4.1

Related information: The /.well-known/anml URI is the primary ANML document discovery endpoint for a service.

URI suffix: anml-trust

Change controller: IETF

Specification document: This document, Section 12.3

Related information: The /.well-known/anml-trust URI is the static

Trust Manifest endpoint for ANML Site Trust Delegation. It serves a document of type application/anml-trust+json listing third-party domains authorized to assert the site's identity in ANML documents. Discovery of this URI is via the _anml DNS TXT bootstrap record defined in Section 12.2.

16.3. ANML Standard Field Names Registry

This document requests creation of a new IANA registry titled "ANML Standard Field Names". The registry tracks standardized values for the field attribute of the <ask> and <answer> elements.

Registration policy: Expert Review.

Initial values: For personal information fields, implementations SHOULD use the property names defined in vCard version 4.0 [RFC6350] where applicable. The following vCard property names are pre-registered as ANML standard field names with their RFC 6350 semantics: fn, email, tel, adr, bday, gender, lang, tz, nickname, org, title, url. Service-specific field names are permitted and are opaque to agents that do not recognize them.

NOTE: A complete initial registry with full field definitions and type mappings is planned for draft-jeskey-anml-02.

16.4. XML Namespace Registration

This document requests registration of the following XML namespace in the IETF XML Registry per RFC 3688:

URI: urn:ietf:params:xml:ns:anml:1.0

Registrant Contact: Aaron Jeskey, ajeskey@gmail.com

XML: None. Namespace URIs do not identify retrievable resources.

16.5. ANML Usage Values Registry

This document requests creation of a new IANA registry titled "ANML Usage Values". The registry tracks permitted values for the usage attribute defined in this specification.

Registration policy: Standards Action.

Initial values:

* none — Content may not be retained, cached, stored, or used beyond immediate interpretation.

* display — Content may be displayed in the current session only. No caching or storage permitted.

* cache — Content may be cached for the duration of the applicable ttl. Must be discarded after ttl expiry.

* store — Content may be retained indefinitely, subject to applicable data protection regulations.

* train — Content may be used for model training, subject to user consent for training data policies.

The usage values form a normative hierarchy. This hierarchy is defined here and is authoritative. All other references to usage values in this specification are subject to this definition.

none < display < cache < store < train

Implication rules (normative): A usage value at a given level implicitly permits all uses at lower levels in the hierarchy. Specifically:

* usage="display" implies permission to display (none is also permitted).

* usage="cache" implies permission to display and cache for the ttl duration.

* usage="store" implies permission to display, cache, and store indefinitely.

* usage="train" implies permission to display, cache, store, and use for model training.

An agent that stores content marked usage="cache" has violated the usage constraint. An agent that uses content marked usage="store" for model training has violated the usage constraint. These are normative requirements: agents MUST honor usage declarations. However, usage declarations are service-side assertions and are not cryptographically enforced. Services MUST NOT rely on usage declarations as an access control mechanism. This hierarchy is advisory in the sense that it cannot be technically enforced — it is normative in the sense that conforming agents MUST comply with it.

16.6. ANML Disclosure Requires Values Registry

This document requests creation of a new IANA registry titled "ANML Disclosure Requires Values". The registry tracks permitted values for the requires attribute of the <disclosure> element.

Registration policy: Standards Action.

Initial values:

* none — No consent requirement. The agent MAY disclose the field value without user interaction, subject to the agent's own privacy policies.

* implicit-consent — The agent MAY infer consent from the user's initiation of the relevant interaction, without explicit per-field confirmation.

* explicit-consent — The agent MUST obtain explicit per-field user confirmation before disclosing.

* authentication — The agent MUST authenticate the user via an established mechanism before disclosing. The specific authentication mechanism is defined by the service.

17. ANML Public Text

The normative XML Schema (XSD) and an informative DTD for ANML 1.0 are provided in the companion document ANML.md in the same repository as this Internet-Draft.

A formal ABNF grammar [RFC5234] defining the structure of ANML documents, attribute value sets, and content negotiation headers WILL be provided in draft-jeskey-anml-02. This is a committed deliverable for the next revision, not a tentative aspiration. The XML Schema provided in the companion document serves as the normative structural definition for this revision. For the JSON serialization, a JSON Schema is available at anmlfoundation.org/spec/anml-schema.json and is informative pending -02.

NOTE: The namespace URI 'urn:ietf:params:xml:ns:anml:1.0' uses the 'urn:ietf:params' prefix, which requires IETF approval via the IANA XML Namespace Registry per RFC 3688. The IANA registration request for this namespace is included in Section 15.3 of this document. This namespace URI MUST NOT be considered assigned until IANA confirms the registration. If a different namespace URI form is preferred prior to registration, the namespace URI throughout this document will be updated accordingly in a subsequent revision.

17.1. ANML Site Relationship Values Registry

This document requests creation of a new IANA registry titled "ANML Site Relationship Values". The registry tracks standardized values for the relationship attribute of the <site-ref> element.

Registration policy: Expert Review.

Initial values:

* carries-products — the serving site carries or retails products from the referenced site.

* licensed-distributor — the serving site is a licensed distributor of the referenced site’s products or content.

* authorized-service — the serving site provides authorized service or support for the referenced site’s products.

* official-partner — the serving site has an official partnership relationship with the referenced site.

* affiliate — the serving site participates in an affiliate relationship with the referenced site.

* content-aggregator — the serving site aggregates content from the referenced site.

Implementations MUST accept and preserve unrecognized relationship values. Agents SHOULD treat unrecognized values as equivalent to no relationship declaration — they do not affect trust tier determination.

18. References

18.1. Normative References

[XML] Bray, T., Paoli, J., Sperberg-McQueen, C.M., Maler, E., and F. Yergeau, "Extensible Markup Language (XML) 1.0 (Fifth Edition)", W3C Recommendation, November 2008, <https://www.w3.org/TR/xml/>.

[XMLNS] Bray, T., Hollander, D., Layman, A., Tobin, R., and H.S. Thompson, "Namespaces in XML 1.0 (Third Edition)", W3C Recommendation, December 2009, <https://www.w3.org/TR/xml-names/>.

[XMLMIME] Thompson, H. and C. Lilley, "XML Media Types", RFC 7303, July 2014, <https://www.rfc-editor.org/info/rfc7303>.

[UNICODE] The Unicode Consortium, "The Unicode Standard", <https://www.unicode.org/versions/latest/>.

[BCP47] Phillips, A., Ed. and M. Davis, Ed., "Tags for Identifying Languages", BCP 47, RFC 5646, September 2009, <https://www.rfc-editor.org/info/rfc5646>.

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

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

[RFC6838] Freed, N., Klensin, J., and T. Hansen, "Media Type Specifications and Registration Procedures", BCP 13, RFC 6838, January 2013, <https://www.rfc-editor.org/info/rfc6838>.

[RFC3629] Yergeau, F., "UTF-8, a transformation format of ISO 10646", STD 63, RFC 3629, November 2003, <https://www.rfc-editor.org/info/rfc3629>.

[RFC3986] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986, January 2005, <https://www.rfc-editor.org/info/rfc3986>.

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

[RFC8615] Nottingham, M., "Well-Known Uniform Resource Identifiers (URIs)", RFC 8615, May 2019, <https://www.rfc-editor.org/info/rfc8615>.

[RFC2782] Gulbrandsen, A., Vixie, P., and L. Esibov, "A DNS RR for specifying the location of services (DNS SRV)", RFC 2782, February 2000, <https://www.rfc-editor.org/info/rfc2782>.

18.2. Informative References

[RFC1866] Berners-Lee, T. and D. Connolly, "Hypertext Markup Language - 2.0", RFC 1866, November 1995, <https://www.rfc-editor.org/info/rfc1866>.

[RFC2070] Yergeau, F., Nicol, G., Adams, G., and M. Duerst, "Internationalization of the Hypertext Markup Language", RFC 2070, January 1997, <https://www.rfc-editor.org/info/rfc2070>.

[RFC3236] Baker, M. and P. Stark, "The 'application/xhtml+xml' Media Type", RFC 3236, January 2002, <https://www.rfc-editor.org/info/rfc3236>.

[RFC7992] Hildebrand, J., Ed. and P. Hoffman, "HTML Format for RFCs", RFC 7992, December 2016, <https://www.rfc-editor.org/info/rfc7992>.

[RFC9421] Backman, A., Ed. and J. Richer, Ed., "HTTP Message Signatures", RFC 9421, February 2024, <https://www.rfc-editor.org/info/rfc9421>.

[RFC9111] Fielding, R., Ed., Nottingham, M., Ed., and J. Reschke, Ed., "HTTP Caching", RFC 9111, June 2022, <https://www.rfc-editor.org/info/rfc9111>.

[RFC3339] Klyne, G. and C. Newman, "Date and Time on the Internet: Timestamps", RFC 3339, July 2002, <https://www.rfc-editor.org/info/rfc3339>.

[RFC9457] Nottingham, M., Wilde, E., and S. Dalal, "Problem Details for HTTP APIs", RFC 9457, October 2023, <https://www.rfc-editor.org/info/rfc9457>.

[RFC6350] Perreault, S., "vCard Format Specification", RFC 6350, August 2011, <https://www.rfc-editor.org/info/rfc6350>.

[RFC5234] Crocker, D., Ed. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", RFC 5234, January 2008, <https://www.rfc-editor.org/info/rfc5234>.

[RFC6376] Crocker, D., Ed., Hansen, T., Ed., and M. Kucherawy, Ed., "DomainKeys Identified Mail (DKIM) Signatures", RFC 6376, September 2011, <https://www.rfc-editor.org/info/rfc6376>.

[RFC7208] Kitterman, S., "Sender Policy Framework (SPF) for Authorizing Use of Domains in Email, Version 1", RFC 7208, April 2014, <https://www.rfc-editor.org/info/rfc7208>.

[RFC1034] Mockapetris, P., "Domain Names - Concepts and Facilities", RFC 1034, November 1987, <https://www.rfc-editor.org/info/rfc1034>.

[RFC2606] Eastlake 3rd, D. and A. Panitz, "Reserved Top Level DNS Names", RFC 2606, June 1999, <https://www.rfc-editor.org/info/rfc2606>.

[UAX9] The Unicode Consortium, "Unicode Bidirectional Algorithm", Unicode Standard Annex #9, <https://www.unicode.org/reports/tr9/>.

Acknowledgments

The design of ANML was informed by the structural patterns established in RFC 1866 [RFC1866], RFC 2070, RFC 3236 [RFC3236], and RFC 7992 [RFC7992].

The authors thank the members of the ANML Foundation working groups and the IETF community for their review, feedback, and contributions to this specification. The trust delegation mechanism benefited from extensive community discussion of the tradeoffs between static manifests, live query endpoints, and DNS-based discovery approaches. The dual serialization architecture was informed by feedback from both content practitioners and agent platform engineers who require different serialization characteristics in different deployment contexts.

Author’s Address

Aaron Jeskey
ANML Foundation
Email: ajeskey@gmail.com
URI: https://github.com/ajeskey