Network Working Group
Internet-Draft
Intended status: Informational
Expires: 22 April 2026

S. Nandakumar
C. Jennings
Cisco

19 October 2025

Selective Disclosure for Agent Discovery and Identity Management (SD-Agent)
draft-nandakumar-agent-sd-jwt-01

Abstract

This document defines how Selective Disclosure for JWTs (SD-JWT) integrates with Agent-to-Agent (A2A) systems to provide privacy-preserving agent discovery and cryptographically verifiable identity management. It specifies the SD-Card format - an SD-JWT encoding of Agent Cards that enables selective disclosure of agent capabilities, contact information, and operational metadata while maintaining cryptographic integrity and preventing correlation across different interaction contexts.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on 22 April 2026.

Copyright Notice

Table of Contents

1. Introduction

The Agent-to-Agent (A2A) specification [A2A-SPEC] defines protocols for autonomous agent discovery and interaction through standardized Agent Cards. Current A2A implementations have following limitations:

1. Information Leakage: Agent Cards expose all capabilities during discovery
2. Weak Authentication: No cryptographic verification of agent identity
3. Static Disclosure: No context-based capability filtering
4. Linkability: Identical presentations enable cross-context tracking

This document specifies SD-Card, an SD-JWT [SD-JWT] encoding of Agent Cards that addresses these limitations through:

* Selective Disclosure: Context-specific capability revelation
* Cryptographic Authentication: Key-bound agent identity verification
* Unlinkable Presentations: Privacy-preserving multi-context interactions
* Backward Compatibility: Interoperability with existing A2A implementations

1.1. Terminology

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.

Agent Card: A JSON metadata document describing an agent’s capabilities, identity, and interaction requirements as defined in [A2A-SPEC].

SD-Card: An Agent Card encoded as an SD-JWT that enables selective disclosure of agent metadata.

Agent Registry: A trusted service that issues and manages SD-Cards for agents within a domain or federation.

Discovery Context: A named security context (e.g., "public", "internal", "diagnostic") that determines which claims are disclosed in an SD-Card presentation.

Capability Disclosure: The selective revelation of specific agent capabilities based on the requesting entity's authorization and context.

Key Binding: Cryptographic proof that the presenter of an SD-Card possesses the private key corresponding to the public key in the SD-Card's cnf claim.

SD-JWT Media Type: The application/vnd.sd-jwt+json media type is used to indicate content that contains Selective Disclosure JWT data within JSON structures. This vendor-specific media type distinguishes SD-JWT presentations from standard JSON content and signals that the receiver should process the content using SD-JWT parsing and verification procedures.

1.2. Notational Conventions

This document follows SD-JWT [SD-JWT] conventions:

* Base64url encoding for binary data
* JWS format for signed tokens
* SHA-256 hashing for selective disclosure
* Tilde (~) separator for SD-JWT serialization:

<Issuer-signed JWT>~<Disclosure 1>~<Disclosure 2>~...<Key Binding JWT>

2. Overview of SD-Card Architecture and Benefits

SD-Card enhances Agent-to-Agent (A2A) interactions by providing privacy-preserving agent discovery and cryptographically verifiable agent authentication. SD-Card enables selective disclosure of agent capabilities, skills, and endpoints based on requester authorization and discovery context, unlinkable agent interactions across different contexts, cryptographic integrity of agent metadata, and key binding for authentication that complements transport-layer security.

Traditional agent discovery in Agent-to-Agent protocol faces several challenges:

Information Leakage: Agent Cards expose all capabilities to any

entity performing discovery, potentially revealing sensitive operational details.

Linkability: Identical capabilities presented to different verifiers enable tracking across contexts.

Authenticity: Without cryptographic verification, Agent Cards can be forged or tampered with during transmission.

Capability Enumeration: Attackers can enumerate all agent capabilities by performing discovery, potentially identifying attack vectors.

Context Insensitivity: Agents cannot adapt capability disclosure based on the requesting entity’s authorization level or intended use case.

Allowing agents to selectively disclose their capabilities addresses these challenges through:

Selective Capability Disclosure: Agents reveal only capabilities relevant to the specific discovery context or requester authorization level.

Unlinkable Presentations: Different presentations of the same agent to different verifiers cannot be linked, enhancing privacy.

Flexible Authentication: Key binding enables strong agent authentication without relying solely on transport security.

Batch Credential Issuance: Multiple context-specific credentials can be issued to the same agent for different use cases.

3. SD-Card Agent Card

This specification defienes "SD-Card". An SD-Card is an Agent Card encoded as an SD-JWT.

3.1. Agent Card Structure

The base Agent Card structure follows the A2A specification [A2A-SPEC] but is extended to support selective disclosure. The card consists of three main components:

1. Base Claims: Always disclosed information that establishes the agent’s identity, protocol version, and basic metadata

2. Selective Disclosure Claims: A2A-specific information that can be selectively revealed based on the discovery context including skills, interfaces, and provider details

3. Cryptographic Binding: Keys and signatures that ensure authenticity and enable secure agent interactions

The Agent Card is represented as a Selective Disclosure JSON Web Token (SD-JWT) that contains both standard A2A Agent Card fields and selective disclosure metadata :


```
{
  "iss": "https://registry.example.com",
  "sub": "agent:georoute-planner-v1",
  "iat": 1704063600,
  "exp": 1735685999,
  "vct": "urn:ietf:params:oauth:token-type:sd-agent-card",
  "_sd_alg": "sha-256",
  "_sd": [
    "kV7i-VgPK9Qj8vYNJ4L8hFG3cR9ZqX2mE5wC6oA1nBs", // skills
    "3sdf8mN2p9uX7L3vE8nGrR5kW6oF4tC9jL7vM2nX8qE", // additionalInterfaces
    "8kF2nM3p4uY9L5cG6rT7qW9oR2vX1jE4mN8sC5fH0aZ", // capabilities
    "9mH4pL5rZ8wQ2nX7cV6sT3kE1oY0jF9uR4vB6nM8fG2", // securitySchemes
    "7kRt2qW5mX9jN4cF7zPsE8nG1vH0oY3lM6xC9fK2bA8", // provider
    "5nK3wL8tB4eR9mY2cF6qG7oT1vX0jH3pN4rS8zE5fC6", // defaultInputModes
    "2pM8fT4nR6oX3cG7zE9qL5wY1vS0jF6kH2nB8mC4rT9"  // defaultOutputModes
  ],
  "protocolVersion": "0.2.9",
  "name": "GeoSpatial Route Planner Agent",
  "description": "Provides advanced route planning and traffic analysis services",
  "version": "1.2.0",
  "url": "https://georoute-agent.example.com/a2a/v1",
  "cnf": {
    "jwk": {
      "kty": "EC",
      "crv": "P-256",
      "x": "TCAER19Zvu3OHF4j4W4vfSVoHIP1ILilDls7vCeGemc",
      "y": "ZxjiWWbZMQGHVWKVQ4hbSIirsVfuecCE6t4jT9F2HZQ"
    }
  }
}
```


The following claims are permanently disclosed in the SD-Card:

iss: The issuer of the Agent Card (typically an Agent Registry). This establishes the trust anchor for the credential and enables signature verification. The issuer MUST be a resolvable URI that provides access to the issuer’s public key material.

sub: The unique identifier of the agent within the issuer's namespace. This identifier MUST be stable across different Agent Card issuances for the same logical agent, enabling correlation when authorized while supporting unlinkable presentations through different salt values.

iat: The time at which the Agent Card was issued (Unix timestamp).

exp: The expiration time of the Agent Card (Unix timestamp).

vct: The verifiable credential type, identifying this as an SD-Card Agent Card. This value MUST be "urn:ietf:params:oauth:token-type:sd-agent-card" for compliance with this specification.

name: The human-readable name of the agent.

description: A brief description of the agent's purpose and primary function. This helps users understand the agent's role without revealing sensitive capability details.

version: The version of the agent software.

cnf: The confirmation key for agent authentication. This public key enables key binding and ensures that only the legitimate agent can present the Agent Card. The key MUST be in JSON Web Key (JWK) format.

3.2. Selective Disclosure Claims

The following claims are selectively disclosed based on the discovery context:

3.2.1. Skills and Capabilities

The skills claim contains an array of A2A AgentSkill objects that the agent can perform. Each skill definition follows the A2A specification and includes input/output modes, examples, and metadata for proper integration.

3.2.1.1. Original Skills JSON Object:

```
"skills": [
  {
    "id": "route-optimizer-traffic",
    "name": "Traffic-Aware Route Optimizer",
    "description": "Calculates optimal driving route between locations with real-time tra
ffic",
    "tags": ["maps", "routing", "navigation", "directions", "traffic"],
    "examples": [
      "Plan a route from '1600 Amphitheatre Parkway, Mountain View, CA' to 'San Francisco
 International Airport' avoiding tolls.",
      "{\"origin\": {\"lat\": 37.422, \"lng\": -122.084}, \"destination\": {\"lat\": 37.7
749, \"lng\": -122.4194}, \"preferences\": [\"avoid_ferries\"]}"
    ],
    "inputModes": ["application/json", "text/plain"],
    "outputModes": [
      "application/json",
      "application/vnd.geo+json",
      "text/html"
    ]
  },
  {
    "id": "custom-map-generator",
    "name": "Personalized Map Generator",
    "description": "Creates custom map images based on user-defined points of interest",
    "tags": ["maps", "customization", "visualization", "cartography"],
    "examples": [
      "Generate a map of my upcoming road trip with all planned stops highlighted.",
      "Show me a map visualizing all coffee shops within a 1-mile radius of my current lo
cation."
    ],
    "inputModes": ["application/json"],
    "outputModes": [
      "image/png",
      "image/jpeg",
      "application/json",
      "text/html"
    ]
  }
]
```


3.2.1.2. SD-JWT Transformation Process:

When converting the skills claim to selective disclosure, the Agent Registry performs the following transformation:

Step 1: Generate Salt Value

A cryptographically secure random salt is generated for the skills claim:

Salt: "_26bc4LT-ac6q2KI6cBW5es"

Step 2: Create Disclosure Array

The skills claim is packaged into a disclosure array format:

```
[
  "_26bc4LT-ac6q2KI6cBW5es",
  "skills",
  [
    {
      "id": "route-optimizer-traffic",
      "name": "Traffic-Aware Route Optimizer",
      "description": "Calculates optimal driving route between locations with real
-time traffic",
      "tags": ["maps", "routing", "navigation", "directions", "traffic"],
      "examples": [
        "Plan a route from '1600 Amphitheatre Parkway, Mountain View, CA' to 'San
Francisco International Airport' avoiding tolls.",
        "{\"origin\": {\"lat\": 37.422, \"lng\": -122.084}, \"destination\": {\"la
t\": 37.7749, \"lng\": -122.4194}, \"preferences\": [\"avoid_ferries\"]}"
      ],
      "inputModes": ["application/json", "text/plain"],
      "outputModes": [
        "application/json",
        "application/vnd.geo+json",
        "text/html"
      ]
    },
    {
      "id": "custom-map-generator",
      "name": "Personalized Map Generator",
      "description": "Creates custom map images based to user-defined points of in
terest",
      "tags": ["maps", "customization", "visualization", "cartography"],
      "examples": [
        "Generate a map of my upcoming road trip with all planned stops highlighte
d.",
        "Show me a map visualizing all coffee shops within a 1-mile radius of my c
urrent location."
      ],
      "inputModes": ["application/json"],
      "outputModes": [
        "image/png",
        "image/jpeg",
        "application/json",
        "text/html"
      ]
    }
  ]
]
```

Step 3: Base64url Encode Disclosure

The disclosure array is JSON-serialized and base64url-encoded:

WyJfMjZiYzRMVC1hYzZxMktJNmNCVzVlcyIsInNraWxscyIsW3siaWQiOiJyb3V0ZS1vcHRp
bWl6ZXItdHJhZmZpYyIsIm5hbWUiOiJUcmFmZmljLUF3YXJlIFJvdXRlIE9wdGltaXplciIs
ImRlc2NyaXB0aW9uIjoiQ2FsY3VsYXRlcyBvcHRpbWFsIGRyaXZpbmcgcm91dGUgYmV0d2Vl
biBsb2NhdGlvbnMgd2l0aCByZWFsLXRpbWUgdHJhZmZpYyIsInRhZ3MiOlsibWFwcyIsInJv
dXRpbmciLCJuYXZpZ2F0aW9uIiwiZGlyZWN0aW9ucyIsInRyYWZmaWMiXSwiZXhhbXBsZXMi
OlsiUGxhbiBhIHJvdXRlIGZyb20gJzE2MDAgQW1waGl0aGVhdHJlIFBhcmt3YXksIE1vdW50
YWluIFZpZXcsIENBJyB0byAnU2FuIEZyYW5jaXNjbyBJbnRlcm5hdGlvbmFsIEFpcnBvcnQn
IGF2b2lkaW5nIHRvbGxiIsIntcIm9yaWdpblwiOiB7XCJsYXRcIjogMzcuNDIyLCBcImxu
Z1wiOiAtMTIyLjA4NH0sIFwiZGVzdGluYXRpb25cIjoge1wibGF0XCI6IDM3Ljc3NDksIFwi
bG5nXCI6IC0xMjIuNDE5NH0sIFwicHJlZmVyZW5jZXNcIjogW1wiYXZvaWRfZmVycmllc1wi
XX0iXSwiaW5wdXRNb2RlcyI6WyJhcHBsaWNhdGlvbi9qc29uIiwidGV4dC9wbGFpbiJdLCJv
dXRwdXRNb2RlcyI6WyJhcHBsaWNhdGlvbi9qc29uIiwiYXBwbGljYXRpb24vdm5kLmdlbytq
c29uIiwidGV4dC9odG1sIl19LHsiaWQiOiJjdXN0b20tbWFwLWdlbmVyYXRvciIsIm5hbWUi
OiJQZXJzb25hbGl6ZWQgTWFwIEdlbmVyYXRvciIsImRlc2NyaXB0aW9uIjoiQ3JlYXRlcyBj
dXN0b20gbWFwIGltYWdlcyBiYXNlZCBvbiB1c2VyLWRlZmluZWQgcG9pbnRzIG9mIGludGVy
ZXN0IiwidGFncyI6WyJtYXBzIiwiY3VzdG9taXphdGlvbiIsInZpc3VhbGl6YXRpb24iLCJj
YXJ0b2dyYXBoeSJdLCJleGFtcGxlcyI6WyJHZW5lcmF0ZSBhIG1hcCBvZiBteSB1cGNvbWlu
ZyByb2FkIHRyaXAgd2l0aCBhbGwgcGxhbm5lZCBzdG9wcyBoaWdobGlnaHRlZC4iLCJTaG93
IG1lIGEgbWFwIHZpc3VhbGl6aW5nIGFsbCBjb2ZmZWUgc2hvcHMgd2l0aGluIGEgc2hvcnQg
IHJhZGl1cyBvZiBteSBjdXJyZW50IGxvY2F0aW9uLiJdLCJpbnB1dE1vZGVzIjpbImFwcGxp
Y2F0aW9uL2pzb24iXSwib3V0cHV0TW9kZXMiOlsiaW1hZ2UvcG5nIiwiaW1hZ2UvanBlZyIs
ImFwcGxpY2F0aW9uL2pzb24iLCJ0ZXh0L2h0bWwiXX1dXQ

Step 4: Calculate SHA-256 Digest

A SHA-256 hash is computed over the base64url-encoded disclosure:

```
digest = SHA256(base64url_encode(disclosure_array))
       = "kV7i-VgPK9Qj8vYNJ4L8hFG3cR9ZqX2mE5wC6oA1nBs"
```

Step 5: Include Digest in SD-JWT

The digest is included in the _sd array of the Issuer-signed JWT payload:

```
    {
      "iss": "https://registry.example.com",
      "sub": "agent:georoute-planner-v1",
      "_sd_alg": "sha-256",
      "_sd": [
        "kV7i-VgPK9Qj8vYNJ4L8hFG3cR9ZqX2mE5wC6oA1nBs",
        "7iF2sM8kT5nL9pX3cV6zR4qW1eY8oU0jH3vB7nM2kF9"
      ],
      "name": "GeoSpatial Route Planner Agent",
      "cnf": { }
    }
```

Step 6: Selective Disclosure

When presenting the agent card, the skills disclosure is included only if authorized:

```
<Issuer-signed JWT>~<Skills Disclosure>~<Other Disclosures>~<Key Binding JWT>
```

During verification, the verifier:

1. Decodes the skills disclosure from base64url

2. Calculates SHA-256 hash of the decoded disclosure

3. Confirms the hash matches the digest in the _sd array

4. Extracts the skills claim value for use

3.2.2. Additional Agent Interfaces

The additionalInterfaces claim defines alternative transport mechanisms and endpoints for A2A communication as specified in the AgentInterface object structure.

3.2.2.1. Original values

The original additionalInterfaces claim in the agent metadata contains:

```
{
  "additionalInterfaces": [
    {
      "url": "https://georoute-agent.example.com/a2a/v1",
      "transport": "JSONRPC"
    },
    {
      "url": "https://georoute-agent.example.com/a2a/grpc",
      "transport": "GRPC"
    },
    {
      "url": "https://georoute-agent.example.com/a2a/json",
      "transport": "HTTP+JSON"
    }
  ]
}
```

3.2.2.2. SD-JWT Transformation Process:

When converting the additionalInterfaces claim to selective disclosure, the Agent Registry performs the following transformation:

Step 1: Generate Salt Value A cryptographically secure random salt is generated for the additionalInterfaces claim:

Salt: "_26bc4LT-ac6q2KI6cBW5es"

Step 2: Create Disclosure Array The additionalInterfaces claim is packaged into a disclosure array format:

```
[
  "_26bc4LT-ac6q2KI6cBW5es",
  "additionalInterfaces",
  [
    {
      "url": "https://georoute-agent.example.com/a2a/v1",
      "transport": "JSONRPC"
    },
    {
      "url": "https://georoute-agent.example.com/a2a/grpc",
      "transport": "GRPC"
    },
    {
      "url": "https://georoute-agent.example.com/a2a/json",
      "transport": "HTTP+JSON"
    }
  ]
]
```

Step 3: Base64url Encode Disclosure

The disclosure array is JSON-serialized and base64url-encoded:

```
WyJfMjZiYzRMVC1hYzZxMktJNmNCVzVlcyIsImFkZGl0aW9uYWxJbnRlcmZhY2VzIixbeyJ1cmwiOiJodHRwczovL
2dlb3JvdXRlLWFnZW50LmV4YW1wbGUuY29tL2EyYS92MSIsInRyYW5zcG9ydCI6IkpTT05SUEMifSx7InVybCI6Im
h0dHBzOi8vZ2Vvcm91dGUtYWdlbnQuZXhhbXBsZS5jb20vYTJhL2dycGMiLCJ0cmFuc3BvcnQiOiJHUlBDIn0seyJ
1cmwiOiJodHRwczovL2dlb3JvdXRlLWFnZW50LmV4YW1wbGUuY29tL2EyYS9qc29uIiwidHJhbnNwb3J0IjoiSFRU
UCtKU09OIn1dXQ
```

Step 4: Calculate SHA-256 Digest

A SHA-256 hash is computed over the base64url-encoded disclosure:

```
digest = SHA256(base64url_encode(disclosure_array))
       = "3sdf8mN2p9uX7L3vE8nGrR5kW6oF4tC9jL7vM2nX8qE"
```

Step 5: Include Digest in SD-JWT

The digest is included in the _sd array of the Issuer-signed JWT payload:

```
{
  "iss": "https://registry.example.com",
  "sub": "agent:georoute-planner-v1",
  "_sd_alg": "sha-256",
  "_sd": [
    "3sdf8mN2p9uX7L3vE8nGrR5kW6oF4tC9jL7vM2nX8qE"
  ]
}
```

3.2.3. Agent Capabilities

The capabilities claim provides information about the agent's A2A protocol feature support as defined in the AgentCapabilities object.

3.2.3.1. Original values

The original capabilities claim in the agent metadata contains:

```
{
  "capabilities": {
    "streaming": true,
    "pushNotifications": true,
    "stateTransitionHistory": false,
    "extensions": [
      {
        "name": "advanced-routing",
        "version": "1.0",
        "description": "Enhanced route optimization with machine learning"
      }
    ]
  }
}
```

3.2.3.2. SD-JWT Transformation Process:

When converting the capabilities claim to selective disclosure, the Agent Registry performs the following transformation:

Step 1: Generate Salt Value

A cryptographically secure random salt is generated for the capabilities claim:

Salt: "_26bc4LT-ac6q2KI6cBW5es"

Step 2: Create Disclosure Array

The capabilities claim is packaged into a disclosure array format:

```
[
  "_26bc4LT-ac6q2KI6cBW5es",
  "capabilities",
  {
    "streaming": true,
    "pushNotifications": true,
    "stateTransitionHistory": false,
    "extensions": [
      {
        "name": "advanced-routing",
        "version": "1.0",
        "description": "Enhanced route optimization with machine learning"
      }
    ]
  }
]
```

Step 3: Base64url Encode Disclosure

The disclosure array is JSON-serialized and base64url-encoded:

```
WyJfMjZiYzRMVC1hYzZxMktJNmNCVzVlcyIsImNhcGFiaWxpdGllcyIseyJzdHJlYW1pbmciOnRydWUsInB1c2hOb
3RpZmljYXRpb25zIjp0cnVlLCJzdGF0ZVRyYW5zaXRpb25IaXN0b3J5IjpmYWxzZSwiZXh0ZW5zaW9ucyI6W3sibm
FtZSI6ImFkdmFuY2VkLXJvdXRpbmciLCJ2ZXJzaW9uIjoiMS4wIiwiZGVzY3JpcHRpb24iOiJFbmhhbmNlZCByb3V
0ZSBvcHRpbWl6YXRpb24gd2l0aCBtYWNoaW5lIGxlYXJuaW5nIn1dfV0
```

Step 4: Calculate SHA-256 Digest

A SHA-256 hash is computed over the base64url-encoded disclosure:

```
digest = SHA256(base64url_encode(disclosure_array))
       = "8kF2nM3p4uY9L5cG6rT7qW9oR2vX1jE4mN8sC5fH0aZ"
```

Step 5: Include Digest in SD-JWT

The digest is included in the _sd array of the Issuer-signed JWT payload:

```
{
  "iss": "https://registry.example.com",
  "sub": "agent:georoute-planner-v1",
  "_sd_alg": "sha-256",
  "_sd": [
    "8kF2nM3p4uY9L5cG6rT7qW9oR2vX1jE4mN8sC5fH0aZ"
  ]
}
```

3.2.4. Security Schemes and Authentication

The securitySchemes claim defines authentication requirements following A2A SecurityScheme format.

3.2.4.1. Original values

The original securitySchemes claim in the agent metadata contains:

```
{
  "securitySchemes": {
    "google": {
      "type": "openIdConnect",
      "openIdConnectUrl": "https://accounts.google.com/.well-known/openid-configuratio
n"
    },
    "apiKey": {
      "type": "apiKey",
      "name": "X-API-Key",
      "in": "header"
    }
  }
}
```

3.2.4.2. SD-JWT Transformation Process:

When converting the securitySchemes claim to selective disclosure, the Agent Registry performs the following transformation:

Step 1: Generate Salt Value

A cryptographically secure random salt is generated for the securitySchemes claim:

Salt: "_26bc4LT-ac6q2KI6cBW5es"

Step 2: Create Disclosure Array

The securitySchemes claim is packaged into a disclosure array format:

```
   [
     "_26bc4LT-ac6q2KI6cBW5es",
     "securitySchemes",
     {
       "google": {
         "type": "openIdConnect",
         "openIdConnectUrl": "https://accounts.google.com/.well-known/openid-configuratio
n"
       },
       "apiKey": {
         "type": "apiKey",
         "name": "X-API-Key",
         "in": "header"
       }
     }
   ]
```

Step 3: Base64url Encode Disclosure

The disclosure array is JSON-serialized and base64url-encoded:

```
WyJfMjZiYzRMVC1hYzZxMktJNmNCVzVlcyIsInNlY3VyaXR5U2NoZW1lcyIseyJnb29nbGUiOnsidHlwZSI6Im9wZ
W5JZENvbm5lY3QiLCJvcGVuSWRDb25uZWN0VXJsIjoiaHR0cHM6Ly9hY2NvdW50cy5nb29nbGUuY29tLy53ZWxsLW
tub3duL29wZW5pZC1jb25maWd1cmF0aW9uIn0sImFwaUtleSI6eyJ0eXBlIjoiYXBpS2V5IiwibmFtZSI6IlgtQVB
JLUtleSIsImluIjoiaGVhZGVyIn19XQ
```

Step 4: Calculate SHA-256 Digest

A SHA-256 hash is computed over the base64url-encoded disclosure:

```
digest = SHA256(base64url_encode(disclosure_array))
       = "9mH4pL5rZ8wQ2nX7cV6sT3kE1oY0jF9uR4vB6nM8fG2"
```

Step 5: Include Digest in SD-JWT

The digest is included in the _sd array of the Issuer-signed JWT payload:

```
{
  "iss": "https://registry.example.com",
  "sub": "agent:georoute-planner-v1",
  "_sd_alg": "sha-256",
  "_sd": [
    "9mH4pL5rZ8wQ2nX7cV6sT3kE1oY0jF9uR4vB6nM8fG2"
  ],
  "security": [
    {"google": ["openid", "profile", "email"]},
    {"apiKey": []}
  ]
}
```

3.2.5. Provider Information

The provider claim contains A2A AgentProvider information about the organization that operates the agent.

3.2.5.1. Original values

The original provider claim in the agent metadata contains:

```
{
  "provider": {
    "organization": "Example Geo Services Inc.",
    "url": "https://www.examplegeoservices.com"
  }
}
```

3.2.5.2. SD-JWT Transformation Process:

When converting the provider claim to selective disclosure, the Agent Registry performs the following transformation:

Step 1: Generate Salt Value

A cryptographically secure random salt is generated for the provider claim:

Salt: "_26bc4LT-ac6q2KI6cBW5es"

Step 2: Create Disclosure Array

The provider claim is packaged into a disclosure array format:

```
[
  "_26bc4LT-ac6q2KI6cBW5es",
  "provider",
  {
    "organization": "Example Geo Services Inc.",
    "url": "https://www.examplegeoservices.com"
  }
]
```

Step 3: Base64url Encode Disclosure

The disclosure array is JSON-serialized and base64url-encoded:

```
WyJfMjZiYzRMVC1hYzZxMktJNmNCVzVlcyIsInByb3ZpZGVyIix7Im9yZ2FuaXphdGlvbiI6IkV4YW1wbGUgR2VvI
FNlcnZpY2VzIEluYy4iLCJ1cmwiOiJodHRwczovL3d3dy5leGFtcGxlZ2Vvc2VydmljZXMuY29tIn1d
```

Step 4: Calculate SHA-256 Digest

A SHA-256 hash is computed over the base64url-encoded disclosure:

```
digest = SHA256(base64url_encode(disclosure_array))
       = "7kRt2qW5mX9jN4cF7zPsE8nG1vH0oY3lM6xC9fK2bA8"
```

Step 5: Include Digest in SD-JWT

The digest is included in the _sd array of the Issuer-signed JWT payload:

```
{
  "iss": "https://registry.example.com",
  "sub": "agent:georoute-planner-v1",
  "_sd_alg": "sha-256",
  "_sd": [
    "7kRt2qW5mX9jN4cF7zPsE8nG1vH0oY3lM6xC9fK2bA8"
  ]
}
```

3.2.6. Input and Output Modes

The defaultInputModes and defaultOutputModes claims specify supported content types for A2A message interactions.

3.2.6.1. Original values

The original defaultInputModes and defaultOutputModes claims in the agent metadata contain:

```
{
  "defaultInputModes": ["application/json", "text/plain"],
  "defaultOutputModes": ["application/json", "image/png"]
}
```

3.2.6.2. SD-JWT Transformation Process:

When converting the defaultInputModes and defaultOutputModes claims to selective disclosure, the Agent Registry performs the following transformation:

Step 1: Generate Salt Value

A cryptographically secure random salt is generated for the input/output modes claims:

Salt: "_26bc4LT-ac6q2KI6cBW5es"

Step 2: Create Disclosure Arrays

Each claim is packaged into separate disclosure array formats:

3.2.6.3. Disclosure for defaultInputModes:

```
[
  "_26bc4LT-ac6q2KI6cBW5es",
  "defaultInputModes",
  ["application/json", "text/plain"]
]
```

3.2.6.4. Disclosure for defaultOutputModes:

```
[
  "_26bc4LT-ac6q2KI6cBW5es",
  "defaultOutputModes",
  ["application/json", "image/png"]
]
```

Step 3: Base64url Encode Disclosures

The disclosure arrays are JSON-serialized and base64url-encoded:

3.2.6.5. defaultInputModes encoded:

```
WyJfMjZiYzRMVC1hYzZxMktJNmNCVzVlcyIsImRlZmF1bHRJbnB1dE1vZGVzIixbImFwcGxpY2F0aW9uL2pzb24iL
CJ0ZXh0L3BsYWluIl1d
```

3.2.6.6. defaultOutputModes encoded:

```
WyJfMjZiYzRMVC1hYzZxMktJNmNCVzVlcyIsImRlZmF1bHRPdXRwdXRNb2RlcyIsWyJhcHBsaWNhdGlvbi9qc29uI
iwiaW1hZ2UvcG5nIl1d
```

Step 4: Calculate SHA-256 Digests

SHA-256 hashes are computed over the base64url-encoded disclosures:

```
digest_input = SHA256(base64url_encode(defaultInputModes_disclosure))
             = "5nK3wL8tB4eR9mY2cF6qG7oT1vX0jH3pN4rS8zE5fC6"

digest_output = SHA256(base64url_encode(defaultOutputModes_disclosure))
              = "2pM8fT4nR6oX3cG7zE9qL5wY1vS0jF6kH2nB8mC4rT9"
```

Step 5: Include Digests in SD-JWT

The digests are included in the _sd array of the Issuer-signed JWT payload:

```
{
  "iss": "https://registry.example.com",
  "sub": "agent:georoute-planner-v1",
  "_sd_alg": "sha-256",
  "_sd": [
    "5nK3wL8tB4eR9mY2cF6qG7oT1vX0jH3pN4rS8zE5fC6",
    "2pM8fT4nR6oX3cG7zE9qL5wY1vS0jF6kH2nB8mC4rT9"
  ]
}
```

3.2.7. Documentation and Support

Additional A2A Agent Card metadata for documentation and extended capabilities.

```
"iconUrl": "https://georoute-agent.example.com/icon.png",
"documentationUrl": "https://docs.examplegeoservices.com/georoute-agent/api",
"supportsAuthenticatedExtendedCard": true
```

3.3. Agent Card Issuance

Agent Cards are issued by trusted Agent Registries, the details of the which are out of scope of this specification

3.3.1. Issuance Process Overview

1. Agent Registration: The agent registers with the Agent Registry, providing its capabilities, endpoints, and authentication information. This includes submitting detailed metadata about the agent's functions, operational requirements, and security capabilities.

2. Identity Verification: The Agent Registry verifies the agent's identity and authorization to claim the specified capabilities.

3. Agent Card Creation: The Agent Registry creates an SD-JWT Agent Card containing the agent's metadata with appropriate claims marked for selective disclosure. The registry determines which claims should be selectively disclosable based on authorization, privacy policies and security requirements.

4. Batch Issuance: Multiple Agent Cards with different disclosure configurations may be issued for different discovery contexts. This enables the same agent to interact in various contexts while maintaining privacy and unlinkability.

Example issuance request based on HTTP POST method with bearing token is shown below:

```
POST /agents/register HTTP/1.1
Host: registry.example.com
Content-Type: application/json
Authorization: Bearer <registry_auth_token>

{
  "agent_id": "ai-assistant-v2",
  "name": "AI Assistant Agent",
  "description": "General purpose AI assistant",
  "version": "2.1.0",
  "public_key": {
    "kty": "EC",
    "crv": "P-256",
    "x": "TCAER19Zvu3OHF4j4W4vfSVoHIP1ILilDls7vCeGemc",
    "y": "ZxjiWWbZMQGHVWKVQ4hbSIirsVfuecCE6t4jT9F2HZQ"
  },
  "disclosure_contexts": [
    {
      "context": "public",
      "disclose": ["skills", "endpoints", "operational_info"]
    },
    {
      "context": "internal",
      "disclose": ["skills", "endpoints", "operational_info",
                   "provider", "security_context"]
    },
    {
      "context": "diagnostic",
      "disclose": ["operational_info", "provider"]
    }
  ]
}
```

4. Agent Discovery Protocols with Privacy Protection

This section describes the protocols and mechanisms for discovering agents:

4.1. Well-Known URI Discovery

The well-known URI discovery mechanism is extended to support SD-JWT Agent Cards:

```
GET /.well-known/agent.json HTTP/1.1
Host: agent.example.com
Accept: application/vnd.sd-jwt+json
```

The application/vnd.sd-jwt+json media type indicates that the client accepts responses containing Selective Disclosure JWT data formatted as JSON. In the examples below, the "agent_card" field contains an SD-JWT formatted according to the Agent Card Structure defined in Section 3.1.

The response contains an SD-JWT Agent Card with context-appropriate disclosures.

```
HTTP/1.1 200 OK
Content-Type: application/vnd.sd-jwt+json

{
  "agent_card": "eyJhbGciOiJFUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiJod...~WyJzYWx0I
iwibmFtZSIsIkFJIEFzc2lzdGFudCJd~WyJzYWx0MiIsImVuZHBvaW50cyIsW3sibmFtZSI6InByaW1h
cnkiLCJ1cmwiOiJodHRwczovL2FnZW50LmV4YW1wbGUuY29tL2FwaS92MSJ9XV0~"
}
```

For privacy-sensitive discovery, the agent MAY require authentication before revealing the Agent Card:

```
GET /.well-known/agent.json HTTP/1.1
Host: agent.example.com
Authorization: Bearer <discovery_token>
Accept: application/vnd.sd-jwt+json
```

4.2. Registry-Based Discovery

Agent Registries support discovery with context-based selective disclosure. In the example below, an agent is discovered based on the skill "route-optimizer-traffic".

```
POST /agents/discover HTTP/1.1
Host: registry.example.com
Content-Type: application/json
Authorization: Bearer <registry_access_token>

{
  "query": {
    "skills": ["route-optimizer-traffic"],
    "availability": "24/7"
  },
  "context": "public",
  "max_results": 10
}
```

4.3. Contextual Discovery

Discovery can be performed with different contexts to control the level of information disclosure:

Public Context: Minimal information suitable for public discovery, including basic skills and public endpoints.

Internal Context: Additional operational details for internal organizational use.

Diagnostic Context: Detailed technical information for troubleshooting and monitoring.

Federation Context: Inter-organizational discovery with appropriate trust relationships.

The same agent may present different Agent Cards for different contexts while maintaining unlinkability between presentations.

5. Cryptographic Authentication and Capability-Based Authorization

SD-Card provides enhanced authentication and authorization mechanisms through SD-JWT integration with cryptographic key binding. This section details the complete authentication flow, key binding mechanisms, and capability-based authorization processes that ensure secure agent interactions.

5.1. SD-JWT with Key Binding

Selective Disclosure JWT (SD-JWT) with Key Binding is a security mechanism that enables privacy-preserving authentication.

An SD-JWT consists of three main parts separated by tildes (~):

1. Issuer-Signed JWT: Contains the base claims and selective disclosure metadata

2. Disclosure Arrays: Base64url-encoded arrays containing the salt and claim data

3. Key Binding JWT (KB-JWT): Proves possession of the confirmation key

```
SD-JWT Format:
<Issuer-Signed JWT>~<Disclosure 1>~<Disclosure 2>~...~<Key Binding JWT>
```

Key binding cryptographically proves that the presenter of an SD-JWT possesses the private key corresponding to the public key specified in the cnf (confirmation) claim of the SD-JWT. This mechanism:

1. Selective Disclosure: Only relevant claims are revealed to the verifier

2. Cryptographic Binding: Proof of possession of a private key

3. Unlinkable Presentations: Different interactions cannot be correlated

4. Replay Protection: Each presentation is unique and time-bound

5.2. Agent Authentication Process

The agent authentication process involves multiple cryptographic steps to ensure secure and verifiable identity establishment:

5.2.1. Step 1: Agent Card Presentation

The agent presents its SD-JWT Agent Card with context-appropriate selective disclosures. The "agent_card" field contains an SD-JWT generated from the following raw input structure:

Raw Input Structure:

The agent_card field is generated from the following base agent information:

```
{
  "iss": "https://registry.example.com",
  "sub": "agent:ai-assistant-v2",
  "iat": 1704063600,
  "exp": 1735685999,
  "vct": "urn:ietf:params:oauth:token-type:sd-a2a-agent-card",
  "_sd_alg": "sha-256",
  "_sd": [
    "kV7i-VgPK9Qj8vYNJ4L8hFG3cR9ZqX2mE5wC6oA1nBs", // skills digest
    "3sdf8mN2p9uX7L3vE8nGrR5kW6oF4tC9jL7vM2nX8qE"  // endpoints digest
  ],
  "name": "AI Assistant Agent",
  "description": "General purpose AI assistant",
  "version": "2.1.0",
  "cnf": {
    "jwk": {
      "kty": "EC",
      "crv": "P-256",
      "x": "TCAER19Zvu3OHF4j4W4vfSVoHIP1ILilDls7vCeGemc",
      "y": "ZxjiWWbZMQGHVWKVQ4hbSIirsVfuecCE6t4jT9F2HZQ"
    }
  }
}
```

Selective Disclosures:

* skills: ["salt1","skills",[{"name":"text-generation"}]]

* endpoints: ["salt2","endpoints",[{"name":"primary"}]]

Resulting SD-JWT Presentation:


```
{
  "agent_card": "eyJhbGciOiJFUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiJodHRwczovL3JlZ2lzdHJ5LmV
4YW1wbGUuY29tIiwic3ViIjoiYWdlbnQ6YWktYXNzaXN0YW50LXYyIiwiaWF0IjoxNzA0MDYzNjAwLCJleHAiOjE3
MzU2ODU5OTksInZjdCI6InVybjppZXRmOnBhcmFtczpvYXV0aDp0b2tlbi10eXBlOnNkLWEyYS1hZ2VudC1jYXJkI
iwiX3NkX2FsZyI6InNoYS0yNTYiLCJfc2QiOltdLCJuYW1lIjoiQUkgQXNzaXN0YW50IEFnZW50IiwiZGVzY3JpcH
Rpb24iOiJHZW5lcmFsIHB1cnBvc2UgQUkgYXNzaXN0YW50IiwidmVyc2lvbiI6IjIuMS4wIiwiY25mIjp7Imp3ayI
6eyJrdHkiOiJFQyIsImNydiI6IlAtMjU2IiwieCI6IlRDQUVSMTladnUzT0hGNGo0VzR2ZlNWb0hJUDFJTGlsRGxz
N3ZDZUdlbWMiLCJ5IjoiWnhqaVdXYlpNUUdIVldLVlE0aGJTSWlyc1ZmdWVjQ0U2dDRqVDlGMkhaUSJ9fX0.signa
ture~WyJzYWx0MSIsInNraWxscyIsW3sibmFtZSI6InRleHQtZ2VuZXJhdGlvbiJ9XV0~WyJzYWx0MiIsImVuZHBv
aW50cyIsW3sibmFtZSI6InByaW1hcnkifV1d~eyJhbGciOiJFUzI1NiIsInR5cCI6ImtiK2p3dCJ9.eyJpYXQiOjE
3MDQwNjM2MDAsImF1ZCI6aWNsaWVudC5leGFtcGxlLmNvbSIsInNkX2hhc2giOiJLdDdRcU5aQnpxbkpLVlE0aGJT
SWlyc1ZmdWVjQ0U2dDRqVDlGMkhaUSIsImludGVyYWN0aW9uX2lkIjoiMTIzNDU2NzgtMTIzNC0xMjM0LTEyMzQtM
TIzNDU2Nzg5YWJjIn0.kb_signature"
}
```


5.2.2. Step 2: Key Binding JWT Creation

The agent creates a Key Binding JWT (KB-JWT) to prove possession of the private key. The KB-JWT structure includes:

KB-JWT Header:

```
{
  "alg": "ES256",
  "typ": "kb+jwt"
}
```

KB-JWT Payload:

```
{
  "iat": 1704063600,
  "aud": "client.example.com",
  "sd_hash": "Kt7QqNZBzqnJKVQ4hbSIirsVfuecCE6t4jT9F2HZQ",
  "interaction_id": "12345678-1234-1234-1234-123456789abc",
  "nonce": "random-nonce-value-12345"
}
```

where,

* iat: Issued at time - prevents replay attacks

* aud: Audience - identifies the intended recipient

* sd_hash: Hash of the SD-JWT presentation (binds KB-JWT to specific presentation)

* interaction_id: Unique interaction identifier - prevents cross-session attacks

* nonce: Optional random value - adds additional replay protection

5.2.3. Step 3: Cryptographic Verification

The client performs comprehensive verification:

1. Agent Card Signature Verification: Validates the issuer's signature

2. Disclosure Verification: Confirms disclosed claims match their digests

3. Key Binding Verification: Validates the KB-JWT signature using the agent's public key

4. Temporal Validation: Checks expiration and freshness of timestamps

5. Audience Validation: Ensures the KB-JWT is intended for this client

5.3. Advanced Authorization Flows

TODO: Add text on how a query can be performed based on Agent's capabilities various authorization contexts, such as public, internal, diagnostic, and federation contexts, roles base discovery.

6. Complete SD-JWT and Key Binding Examples

This section provides comprehensive examples of SD-JWT transformations and key binding processes for A2A Agent Cards, demonstrating input formats, selective disclosure transformations, and the complete authentication flow.

Implementation Note: Complete source code implementing all examples in this specification can be found in the examples/ directory of the repository.

6.1. SD-JWT Agent Card Creation Process

6.1.1. Step 1: Original A2A Agent Card (Input)

Starting with a complete A2A Agent Card as defined in the specification:


```
{
  "protocolVersion": "0.2.9",
  "name": "GeoSpatial Route Planner Agent",
  "description": "Provides advanced route planning, traffic analysis, and custom map g
eneration services",
  "url": "https://georoute-agent.example.com/a2a/v1",
  "preferredTransport": "JSONRPC",
  "additionalInterfaces": [
    {"url": "https://georoute-agent.example.com/a2a/v1", "transport": "JSONRPC"},
    {"url": "https://georoute-agent.example.com/a2a/grpc", "transport": "GRPC"},
    {"url": "https://georoute-agent.example.com/a2a/json", "transport": "HTTP+JSON"}
  ],
  "provider": {
    "organization": "Example Geo Services Inc.",
    "url": "https://www.examplegeoservices.com"
  },
  "iconUrl": "https://georoute-agent.example.com/icon.png",
  "version": "1.2.0",
  "documentationUrl": "https://docs.examplegeoservices.com/georoute-agent/api",
  "capabilities": {
    "streaming": true,
    "pushNotifications": true,
    "stateTransitionHistory": false
  },
  "securitySchemes": {
    "google": {
      "type": "openIdConnect",
      "openIdConnectUrl": "https://accounts.google.com/.well-known/openid-configuratio
n"
    }
  },
  "security": [{"google": ["openid", "profile", "email"]}],
```

```
     "defaultInputModes": ["application/json", "text/plain"],
     "defaultOutputModes": ["application/json", "image/png"],
     "skills": [
       {
         "id": "route-optimizer-traffic",
         "name": "Traffic-Aware Route Optimizer",
         "description": "Calculates the optimal driving route between two or more locatio
ns",
         "tags": ["maps", "routing", "navigation", "directions", "traffic"],
         "examples": [
           "Plan a route from '1600 Amphitheatre Parkway, Mountain View, CA' to 'San Fran
cisco International Airport' avoiding tolls."
         ],
         "inputModes": ["application/json", "text/plain"],
         "outputModes": ["application/json", "application/vnd.geo+json", "text/html"]
       },
       {
         "id": "custom-map-generator",
         "name": "Personalized Map Generator",
         "description": "Creates custom map images or interactive map views",
         "tags": ["maps", "customization", "visualization", "cartography"],
         "examples": [
           "Generate a map of my upcoming road trip with all planned stops highlighted."
         ],
         "inputModes": ["application/json"],
         "outputModes": ["image/png", "image/jpeg", "application/json", "text/html"]
       }
     ],
     "supportsAuthenticatedExtendedCard": true
   }
```

6.1.2. Step 2: SD-JWT Transformation with Selective Disclosure

The Agent Registry transforms this into an SD-JWT with selective disclosure markers. The Issuer-Signed JWT contains base claims and selective disclosure digests:

6.1.2.1. Issuer-Signed JWT Header:

```
{
  "alg": "ES256",
  "typ": "JWT"
}
```

6.1.2.2. Issuer-Signed JWT Payload:

```
   {
     "iss": "https://registry.example.com",
     "sub": "agent:georoute-planner-v1",
     "iat": 1704063600,
     "exp": 1735685999,
     "vct": "urn:ietf:params:oauth:token-type:sd-agent-card",
     "_sd_alg": "sha-256",
     "_sd": [
       "kV7i-VgPK9Qj8vYNJ4L8hFG3cR9ZqX2mE5wC6oA1nBs",
       "7iF2sM8kT5nL9pX3cV6zR4qW1eY8oU0jH3vB7nM2kF9",
       "3bN8mL5qR7kX9jV2cF6zT4pW1eY8oU0jH3vB7nM2kG1",
       "5eQ2rP9sL3mX7jV2cF6zR4qW1eY8oU0jH3vB7nM2kH8",
       "1gM4oK7pT5nX9jV2cF6zT4pW1eY8oU0jH3vB7nM2kJ6",
       "9cV8jT2mP5rX7kV9jF6zT4pW1eY8oU0jH3vB7nM2kL4",
       "2hR6kW4nT7mX9jV2cF6zT4pW1eY8oU0jH3vB7nM2kN1"
     ],
     "protocolVersion": "0.2.9",
     "name": "GeoSpatial Route Planner Agent",
     "description": "Provides advanced route planning, traffic analysis, and custom map g
eneration services",
     "url": "https://georoute-agent.example.com/a2a/v1",
     "version": "1.2.0",
     "defaultInputModes": ["application/json", "text/plain"],
     "defaultOutputModes": ["application/json", "image/png"],
     "cnf": {
       "jwk": {
         "kty": "EC",
         "crv": "P-256",
         "x": "TCAER19Zvu3OHF4j4W4vfSVoHIP1ILilDls7vCeGemc",
         "y": "ZxjiWWbZMQGHVWKVQ4hbSIirsVfuecCE6t4jT9F2HZQ"
       }
     }
   }
```


6.1.3. Step 3: Disclosure Arrays for Selective Claims

Each selectively disclosable claim is encoded as a separate disclosure array:

6.1.3.1. Disclosure 1 - Skills:


```
["_26bc4LT-ac6q2KI6cBW5es", "skills", [{"id":"route-optimizer-traffic","name":"Traffic
-Aware Route Optimizer","description":"Calculates the optimal driving route between two o
r more locations","tags":["maps","routing","navigation","directions","traffic"],"examples
":["Plan a route from '1600 Amphitheatre Parkway, Mountain View, CA' to 'San Francisco In
ternational Airport' avoiding tolls."],"inputModes":["application/json","text/plain"],"ou
tputModes":["application/json","application/vnd.geo+json","text/html"]},{"id":"custom-map
-generator","name":"Personalized Map Generator","description":"Creates custom map images
or interactive map views","tags":["maps","customization","visualization","cartography"],"
examples":["Generate a map of my upcoming road trip with all planned stops highlighted."]
,"inputModes":["application/json"],"outputModes":["image/png","image/jpeg","application/j
son","text/html"]}]]
```


Base64url encoded:


```
WyJfMjZiYzRMVC1hYzZxMktJNmNCVzVlcyIsInNraWxscyIsW3siaWQiOiJyb3V0ZS1vcHRpbWl6ZXItdHJhZmZpY
yIsIm5hbWUiOiJUcmFmZmljLUF3YXJlIFJvdXRlIE9wdGltaXplciIsImRlc2NyaXB0aW9uIjoiQ2FsY3VsYXRlcy
B0aGUgb3B0aW1hbCBkcml2aW5nIHJvdXRlIGJldHdlZW4gdHdvIG9yIG1vcmUgbG9jYXRpb25zIiwidGFncyI6WyJ
tYXBzIiwicm91dGluZyIsIm5hdmlnYXRpb24iLCJkaXJlY3Rpb25zIiwidHJhZmZpYyJdLCJleGFtcGxlcyI6WyJQ
bGFuIGEgcm91dGUgZnJvbSAnMTYwMCBBbXBoaXRoZWF0cmUgUGFya3dheSwgTW91bnRhaW4gVmlldywgQ0EnIHRvI
CdTYW4gRnJhbmNpc2NvIEludGVybmF0aW9uYWwgQWlycG9ydCcgYXZvaWRpbmcgdG9sbHMuIl0sImlucHV0TW9kZX
MiOlsiYXBwbGljYXRpb24vanNvbiIsInRleHQvcGxhaW4iXSwib3V0cHV0TW9kZXMiOlsiYXBwbGljYXRpb24vanN
```

vbiIsImFwcGxpY2F0aW9uL3ZuZC5nZW8ranNvbiIsInRleHQvaHRtbCJdfSx7ImlkIjoiY3VzdG9tLW1hcC1nZW5lcmF0b3IiLCJuYW1lIjoiUGVyc29uYWxpemVkIE1hcCBHZW5lcmF0b3IiLCJkZXNjcmlwdGlvbiI6IkNyZWF0ZXMgY3VzdG9tIG1hcCBpbWFnZXMgb3IgaW50ZXJhY3RpdmUgbWFwIHZpZXdzIiwidGFncyI6WyJtYXBzIiwiY3VzdG9taXphdGlvbiIsInZpc3VhbGl6YXRpb24iLCJjYXJ0b2dyYXBoeSJdLCJleGFtcGxlcyI6WyJHZW5lcmF0ZSBhIG1hcCBvZiBteSB1cGNvbWluZyByb2FkIHRyaXAgd2l0aCBhbGwgcGxhbm5lZCBzdG9wcyBoaWdobGlnaHRlZC4iXSwiaW5wdXRNb2RlcyI6WyJhcHBsaWNhdGlvbi9qc29uIl0sIm91dHB1dE1vZGVzIjpbImltYWdlL3BuZyIsImltYWdlL2pwZWciLCJhcHBsaWNhdGlvbi9qc29uIiwidGV4dC9odG1sIl19XV1d

6.1.3.2. Disclosure 2 - Additional Interfaces:

["_3sdf8mN-2p9uX7L3vE8nGr", "additionalInterfaces", [{"url":"https://georoute-agent.example.com/a2a/v1","transport":"JSONRPC"},{"url":"https://georoute-agent.example.com/a2a/grpc","transport":"GRPC"},{"url":"https://georoute-agent.example.com/a2a/json","transport":"HTTP+JSON"}]]

6.1.3.3. Disclosure 3 - Provider:

["_7kRt2qW-5mX9jN4cF7zPs", "provider", {"organization":"Example Geo Services Inc.","url":"https://www.examplegeoservices.com"}]

6.1.4. Step 4: Complete SD-JWT Serialization

The complete SD-JWT is serialized (truncated for readability):

eyJhbGciOiJFUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiJodHRwczovL3JlZ2lzdHJ5LmV4YW1wbGUuY29tIiwi
c3ViIjoiYWdlbnQ6Z2Vvcm91dGUtcGxhbm5lci12MSIsImlhdCI6MTcwNDA2MzYwMCwiZXhwIjoxNzM1Njg1OTk5L
CJ2Y3QiOiJ1cm46aWV0ZjpwYXJhbXM6b2F1dGg6dG9rZW4tdHlwZTpzZC1hZ2VudC1jYXJkIiwiX3NkX2FsZyI6In
NoYS0yNTYiLCJfc2QiOlsia1Y3aS1WZ1BLOVI...~WyJfMjZiYzRMVC1hYzZxMktJNmNCVzVlcyIsInNraWxscyIs
W3siaWQiOiJyb3V0ZS1vcHRpbWl6ZXItdHJhZmZpYyIsIm5hbWUiOiJUcmFmZmljLUF3YXJlIFJvdXRlIE9wdGlta
XplciIsImRlc2NyaXB0aW9uIjoiQ2FsY3VsYXRlcyB0aGUgb3B0aW1hbCBkcml2aW5nIHJvdXRlIGJldHdlZW4gdH
dvIG9yIG1vcmUgbG9jYXRpb25zIiwidGFncyI6WyJtYXBzIiwicm91dGluZyIsIm5hdmlnYXRpb24iLCJkaXJlY3R
pb25zIiwidHJhZmZpYyJdLCJleGFtcGxlcyI6WyJQbGFuIGEgcm91dGUgZnJvbSAnMTYwMCBBbXBoaXRoZWF0cmUg
UGFya3dheSwgTW91bnRhaW4gVmlldywgQ0EnIHRvICdTYW4gRnJhbmNpc2NvIEludGVybmF0aW9uYWwgQWlycG9yd
CcgYXZvaWRpbmcgdG9sbHMuIl0sImlucHV0TW9kZXMiOlsiYXBwbGljYXRpb24vanNvbiIsInRleHQvcGxhaW4iXS
wib3V0cHV0TW9kZXMiOlsiYXBwbGljYXRpb24vanNvbiIsImFwcGxpY2F0aW9uL3ZuZC5nZW8ranNvbiIsInRleHQ
vaHRtbCJdfSx7ImlkIjoiY3VzdG9tLW1hcC1nZW5lcmF0b3IiLCJuYW1lIjoiUGVyc29uYWxpemVkIE1hcCBHZW5l
cmF0b3IiLCJkZXNjcmlwdGlvbiI6IkNyZWF0ZXMgY3VzdG9tIG1hcCBpbWFnZXMgb3IgaW50ZXJhY3RpdmUgbWFwI
HZpZXdzIiwidGFncyI6WyJtYXBzIiwiY3VzdG9taXphdGlvbiIsInZpc3VhbGl6YXRpb24iLCJjYXJ0b2dyYXBoeS
JdLCJleGFtcGxlcyI6WyJHZW5lcmF0ZSBhIG1hcCBvZiBteSB1cGNvbWluZyByb2FkIHRyaXAgd2l0aCBhbGwgcGx
hbm5lZCBzdG9wcyBoaWdobGlnaHRlZC4iXSwiaW5wdXRNb2RlcyI6WyJhcHBsaWNhdGlvbi9qc29uIl0sIm91dHB1
dE1vZGVzIjpbImltYWdlL3BuZyIsImltYWdlL2pwZWciLCJhcHBsaWNhdGlvbi9qc29uIiwidGV4dC9odG1sIl19X
V1d~WyJfM3NkZjhtTi0ycDl1WDdMM3ZFOG5HciIsImFkZGl0aW9uYWxJbnRlcmZhY2VzIixbeyJ1cmwiOiJodHRwc
zovL2dlb3JvdXRlLWFnZW50LmV4YW1wbGUuY29tL2EyYS92MSIsInRyYW5zcG9ydCI6IkpTT05SUEMifSx7InVybC
I6Imh0dHBzOi8vZ2Vvcm91dGUtYWdlbnQuZXhhbXBsZS5jb20vYTJhL2dycGMiLCJ0cmFuc3BvcnQiOiJHUUlBDIn0
seyJ1cmwiOiJodHRwczovL2dlb3JvdXRlLWFnZW50LmV4YW1wbGUuY29tL2EyYS9qc29uIiwidHJhbnNwb3J0Ijoi
SFRUUCtKU09OIn1dXQ~WyJfN2tSdDJxVy01bVg5ak40Y0Y3elBzIiwicHJvdmlkZXIiLHsib3JnYW5pemF0aW9uIj
oiRXhhbXBsZSBHZW8gU2VydmljZXMgSW5jLiIsInVybCI6Imh0dHBzOi8vd3d3LmV4YW1wbGVnZW9zZXJ2aWNlcy5
jb20ifV1

7. IANA Considerations

TODO

8. Security Considerations

TODO

{backmatter}

9. References

9.1. Normative References

[A2A-SPEC] Project, A., "Agent-to-Agent Protocol Specification", Version 0.2.5, 2024, <https://a2aproject.github.io/A2A/v0.2.5/specification>.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, 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, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

[SD-JWT] Fett, D., Yasuda, K., and B. Campbell, "Selective Disclosure for JWTs (SD-JWT)", Work in Progress, Internet-Draft, draft-ietf-oauth-selective-disclosure-jwt-22, November 2024, <https://datatracker.ietf.org/doc/html/draft-ietf-oauth-selective-disclosure-jwt-22>.

9.2. Informative References

[RFC6749] Hardt, D., Ed., "The OAuth 2.0 Authorization Framework", RFC 6749, DOI 10.17487/RFC6749, October 2012, <https://www.rfc-editor.org/info/rfc6749>.

[RFC7515] Jones, M., Bradley, J., and N. Sakimura, "JSON Web Signature (JWS)", RFC 7515, DOI 10.17487/RFC7515, May 2015, <https://www.rfc-editor.org/info/rfc7515>.

[RFC7519] Jones, M., Bradley, J., and N. Sakimura, "JSON Web Token (JWT)", RFC 7519, DOI 10.17487/RFC7519, May 2015, <https://www.rfc-editor.org/info/rfc7519>.

Authors' Addresses

Suhas Nandakumar
Cisco
Email: snandaku@cisco.com

Cullen Jennings
Cisco
Email: fluffy@cisco.com