Swefed OIDF Sandbox Federation

Disclaimer

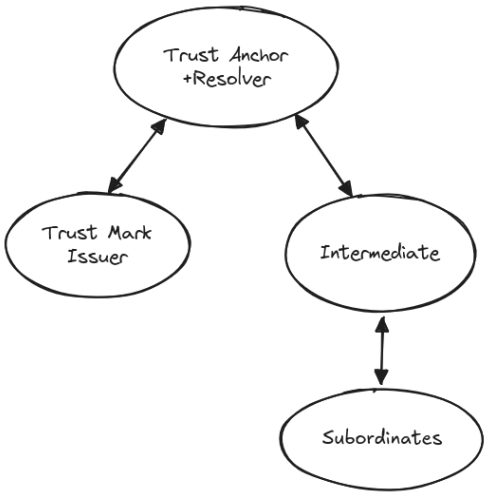
This document provides an overview of the Swefed OIDF Sandbox environment. It focuses on metadata handling, trust chain validation, and Trust Mark usage in the context of OpenID Federation 1.0. For complete details, consult the OpenID Federation 1.0 specification - draft 43.

Introduction

The Swefed OIDF Sandbox is an isolated environment for testing OpenID Federation. It allows Relying Parties, OpenID Providers, and supporting entities to validate interoperability, metadata exchange, and trust chain resolution under a Trust Anchor.

Federation Architecture and Core Entities

The federation has a hierarchical structure with the Trust Anchor as the root of trust. Subordinate entities include Resolvers, Intermediates, Trust Mark Issuers, OpenID Providers, and Relying Parties.



Entities and Their Roles

Trust Anchor: Root of trust. Defines policies and signs metadata.

Resolver: Builds and returns validated trust chains by following authority hints and subordinate statements, allowing entities to establish trust with a Trust Anchor.

Trust Mark Issuer: Issues signed Trust Marks certifying compliance with federation requirements.

Intermediate: Manages subordinate entities and aggregates metadata.

OpenID Provider: Authenticates users and issues tokens under federation policies.

Relying Party: Consumes identity information from OpenID Providers through validated trust chains.

Trust Workflow

Each entity publishes an Entity Configuration at `/.well-known/openid-federation`. This is a signed JWT containing the entity identifier, role-specific metadata, authority hints, and a JWKS by value. Trust Marks may be included.

Validation is done by building a trust chain from the entity to the Trust Anchor. The chain is verified using the Trust Anchor's keys. Trust Marks add assurance of policy compliance.

Trust is established dynamically through metadata exchange and chain resolution, enabling scalable onboarding without static configuration.

Trust Mark Issuance

A Trust Mark Issuer evaluates an entity against defined requirements. If compliant, it issues a signed JWT Trust Mark including the required claims iss (issuer), sub (subject), id (trust mark identifier), iat (issued at), and exp (expiration).

Examples

RP to OP Interaction

1. The Relying Party fetches the OpenID Provider's Entity Configuration.
2. The Relying Party resolves and validates the trust chain using the Resolver to the Trust Anchor.
3. If trust is valid, the Relying Party registers with the OpenID Provider.
4. Authentication and token flows proceed under validated trust.

Fetching Entity Configuration

The following command extracts and displays the payload of an Entity Configuration. It is useful for inspection, but it does **not** validate the JWT signature. Signature validation must always be performed with trusted keys.

```
curl -s https://trust-anchor.oidf.swefed.se/.well-known/openid-federation \
| cut -d '.' -f2 \
| tr '_-' '/+' \
| base64 -d 2>/dev/null \
| jq .
```

Explanation of each step

- **`curl -s`**: fetches the JWT, `-s` silences progress.
- **`cut -d '.' -f2`**: extracts the **payload** from the JWT (middle part).
- **`tr '_-' '/+'`**: translates Base64URL alphabet into standard Base64.
- **`base64 -d`**: decodes the payload.
- **`jq .`**: pretty-prints the JSON.

Nodes

The following base nodes are operated by the federation operator and form the core of the Swefed OIDF Sandbox. These nodes provide the trust anchor, resolution services, and supporting infrastructure.

Additional nodes such as OpenID Providers (OPs), Relying Parties (RPs), and further intermediates are contributed and managed by Sandbox participants.

Trust Anchor

- URL: `https://trust-anchor.oidf.swefed.se`
- Role: Root of trust. Publishes federation policies and signing keys.
- Provides federation endpoints: `fetch`, `list`, `resolve`.

Resolver

- URL: `https://trust-anchor.oidf.swefed.se` (co-located with TA)
- Role: The Resolver builds trust chains using `authority_hints` and validates signatures up to the TA.
- Endpoint: `/resolve`.

Trust Mark Issuer

- URL: `https://trust-mark-issuer.oidf.swefed.se`
- Role: Issues and manages Trust Marks.
- Provides endpoints: `trust_mark`, `trust_mark_list`, `trust_mark_status`.
- Declares the Trust Anchor in `authority_hints`.

Intermediate

- URL: `https://intermediate.oidf.swefed.se`
- Role: Aggregates and distributes metadata.
- Provides federation endpoints: `fetch`, `list`, `resolve`.
- Declares the Trust Anchor in `authority_hints`.

Usage Notes

- All nodes expose their Entity Configuration at `/.well-known/openid-federation`.
- Trust chains must always be validated against the Trust Anchor.
- JWT signatures must be verified with the published keys from trusted entities.
- Trust Marks must be validated with the Trust Mark Issuer's keys, provided the issuer's trust chain resolves to the Trust Anchor.

Entity Integration

This section explains how to connect an entity to the Swefed Sandbox Trust Infrastructure. It covers metadata exposure, configuration of trust anchors, authority hints, and trust marks.

Prerequisites

- Network access from your entity to the Sandbox Trust Anchor
- Externally accessible HTTPS endpoint for your entity

Key Configuration Points

entity Configuration Information

The entity must expose an Entity Configuration at:

`/.well-known/openid-federation`

The configuration endpoint publishes the entity Configuration document, which provides the entity's configuration details for participants in the Trust Infrastructure.

- **Define the Endpoint:** The endpoint is defined under the following path: /.well-known/openid-federation
- **Implementation:** Ensure this endpoint serves the entity's entity Configuration.

Shortened example of an entity Configuration:

```
{
  "sub": "https://my-entity.example.com",
  "authority_hints": [
    "https://my-intermediate.example.org"
  ],
  "metadata": {
    "federation_entity": {
      "organization_name": "Example Org",
      "contacts": ["support@example.com"]
    },
    "oauth_authorization_server": {
      "token_endpoint": "https://my-entity.example.com/token",
      "authorization_endpoint": "https://my-entity.example.com/authorize",
      "jwks_uri": "https://my-entity.example.com/jwks/oauth"
    }
  },
  "jwks": {
    "keys": [
      {
        "kty": "RSA",
        "use": "sig",
        "kid": "example-key-id",
        "e": "AQAB",
        "n": "example-modulus"
      }
    ]
  }
}
```

Trust Anchors

The Trust Anchor is the root of the federation's trust chain. It defines policies and anchors all subordinate chains.

For configuration, you must add **both** the Trust Anchor's entity identifier and its public keys:

- **Entity ID (URL):** https://trust-anchor.oidf.swefed.se
- **Public Keys (JWKS)**:

```
{
    "keys": [
        {
            "kty": "RSA",
            "use": "sig",
            "kid": "JI0SzIoDD4gaEzmLUgKGj-eu5jkTpEHW1JKk7OGdyiQ",
            "n": "jSDBu2-S3Z2-HN2ToJv3zQWe0lizg2VNq4Y68Xy1513nAv1fa9p1Sw5YRyhWUxCUnWnNCmGS5g11-
Npqco8A7XUGzroYgOmD_1Pv0xBhOFpVQDwzLs7vmyaH2GiHDrUGBHVUJrdpjMsT5YD3xw39hQLDVlZ9qZeS4U0jUhGR4qoF5f4KuFxSb6
OykDRvMcP2jSf6fvLw5oKg5b16LqLcAB-
LVLC43QRXDRj1zAUMuuBa4n_dwyRG0aF2FsNsLnwvWLVGTrcSuwdnagl_T8oLLE_X5HtWQyOn0dmS_zbVZnkzxxso9usuMD4f-
7BDTZq5kWRa-HUBqjpEOOcEp-ix-Q",
            "e": "AQAB"
        },
        {
            "kty": "EC",
            "use": "sig",
            "kid": "pj4FG-prDKHyUrliyQb55zFOG3jwu_ivRPWnlB9YgKc",
            "crv": "P-256",
            "x": "XO0BmxPvPM7kfkfXwpXWOpF8Zm_-xwe_lfz6NWz4dxA",
            "y": "tXrhZ592sNhS1h3J79_d9kB-nphMLSUkegJ99ak1s-4"
        }
    ]
}
```

Authority Hints

The authority_hints parameter specifies the URL of the Intermediate Entities or Trust Anchors that are Immediate Superiors of the entity. This helps other Trust Infrastructure participants understand upstream trust relationships.

- **Add to Configuration**: Add authority_hints in your entity's configuration

Registering the Entity as a Subordinate Entity

In the Sandbox Trust Infrastructure, every entity must be registered as a **Subordinate Entity** under a **Superior Entity** (such as a Trust Anchor or an Intermediate). Registration ensures that the entity is formally included in the federation trust hierarchy.

Preparing the Registration Document

To register, you must prepare a JSON document that includes your entity identifier, declared entity types, and public keys.

Start with this template:

```
{
  "<entity-identifier>": {
    "entity_types": [
      "federation_entity",
      "<additional-entity-types>"
    ],
    "jwks": {
      "keys": [
        {
            "<Key 1>"
        },
                {
                        "<Key 2>"
                }
      ]
    }
  }
}
```

Instructions

- Replace `<entity-identifier>` with the **entity_id** of your entity (typically its HTTPS URL).
- Every entity must include **`federation_entity`** as one of its types.
- Add the entity types that apply to your role in the Sandbox:
    - **`openid_relying_party`** for RPs.
    - **`openid_provider`** for OPs.
    - **`oauth_authorization_server`** for entities acting as OAuth 2.0 AS.
- Place your public keys in the "`jwks`" section. Only **public key parameters** are included. Private key material must never be published.

Example

```
{
  "https://entity.example.com": {
    "entity_types": [
      "federation_entity",
      "openid_provider",
      "oauth_authorization_server"
    ],
    "jwks": {
      "keys": [
        {
            "kty": "EC",
            "use": "sig",
            "kid": "cFFvS3F3ZEZkZXFDS3VtamR2WlI2UEFBNG9neTFMdi1JOFlNdkxHODJWOA",
            "crv": "P-256",
            "x": "QFPmIUbY-lTLavyqzT-GqVKHCE28ng5QTWzbC3kMMJ8",
            "y": "vRK1LZjF3DSRqEflUwCf5obg86yWvv2Iekae7A5u35E",
            "d": "W5kLTh8IKEY0U281a7ZmGYbFAzV5kq0SjacTIufKBYM"
        }
    }
  }
}
```

This registration document should be submitted to the Sandbox operator to complete onboarding of the entity.

Trust Marks

Trust Marks are JWTs issued by a Trust Mark Issuer to validate compliance with Trust Infrastructure policies.

Types of Trust Marks

The following Trust Marks are available for issuance:

- **Sambi**:
  - **ID**: https://trust-mark.oidf.swefed.se/sambi

Retrieving Trust Marks

Trust Marks will be supplied on request.

1. **Inputs to Trust Mark Issuer operator**:
   - **`id`**: The identifier for the Trust Mark.
   - **`sub`**: The entity's entity Identifier.
2. **Steps**:
   - Supply the **`id`** and **`sub`** to the Trust Mark Issuer.
   - Retrieve the issued Trust Mark as a signed JWT.
3. **Include in Metadata**: Add issued Trust Marks to your entity's configuration:

Testing Trust Marks

1. **Decode JWT**: Use tools like `jwt.io` to inspect the Trust Mark's claims and ensure all required fields are present.
2. **Verify Signature**: Validate the JWT signature against the Trust Mark Issuer's public key.
3. **Check Expiration**: Ensure the `exp` claim (if present) has not expired.
4. **Validate References**: Follow the `ref` URL (if provided) to confirm compliance with human-readable policy documents.

Steps to Connect the Entity

1. **Configure the entity**:
   - Update the entity's configuration to include `authority_hints`, `trust_marks`, and `trust_anchors`.
2. **Register with the Trust Infrastructure:**
   - Share your **`entity_registration.json`** with the Trust Anchor or superior entity for registration.
3. **Validate Configuration**:
   - Test the entity using testing tools or with other entities sandbox environment.
4. **Monitor the Connection**:
   - Regularly verify the status and ensure Trust Marks are up-to-date.

Testing the Trust Relationships

1. **Validate trust marks**
   Use tools like jwt.io to decode and verify trust marks using the **Trust Mark Issuer's** public keys.
2. **Retrieve metadata**
   Ensure the .well-known/openid-federation endpoint correctly serves the entity's entity configuration:

   ```
   curl https://your-entity.example.com/.well-known/openid-federation
   ```

3. **Check authority hints**
   Verify that `authority_hints` points to the correct Superior
4. **Validate public keys**
   Confirm that the Trust Anchor's public keys match those provided in your local configuration.