OpenID Connect Federation

How to build multilateral federations using OIDC

Roland Hedberg, June 2022

OIDC layers that are affected

1. Provider discovery
2. Dynamic client registration
3. Authorization/Authentication
4. Access Token/Refresh Token
5. Userinfo

Toolbox

- Trusted information
- Metadata policy
- Trust marks
- Federation services
- Client registration methods

Trusted information

- Correct
  - tamper-resistance
  - non-repudiability
  - Rules for metadata
- Based on a trusted 3rd party (trust anchor)
- Expressed as an ordered chain of entity statements

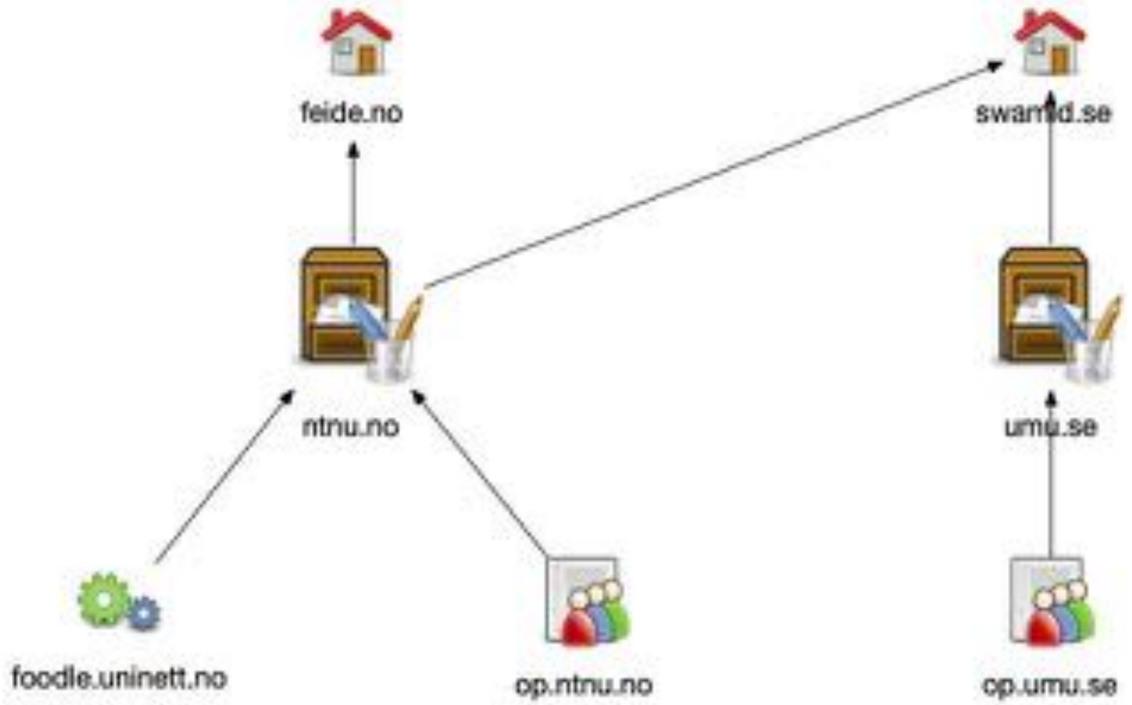
Entity Statement

- Information about an entity
- Can be an entity's view of itself (self-signed entity statement/entity configuration) or a superiors view of a subordinate.
- Signed JWT

```
{
  "iss": ”https://rp.umu.se”,
  "sub": ”https://rp.umu.se”,
  "iat": 1516239022,
  "exp": 1516298022,
  "metadata": {
    "openid_relying_party": {
      "application_type": "web",
      "redirect_uris": [
        ”https://rp.umu.se/rp/callback”
      ],
      "grant_types": [
        "authorization_code",
        "implicit"
      ],
      "jwks_uri": ”https://rp.umu.se/static/jwks.json”
    }
  },
  "jwks": {
    "keys": [
      {
        "kid": "key1",
        "kty": "RSA",
        "use": "sig"
        "e": "AQAB",
        "n": "pnXBOusEANuug6ewezb9J_...",
      }
    ]
  },
  "authority_hints": [
    ”https://federation.umu.se”
  ]
}
```

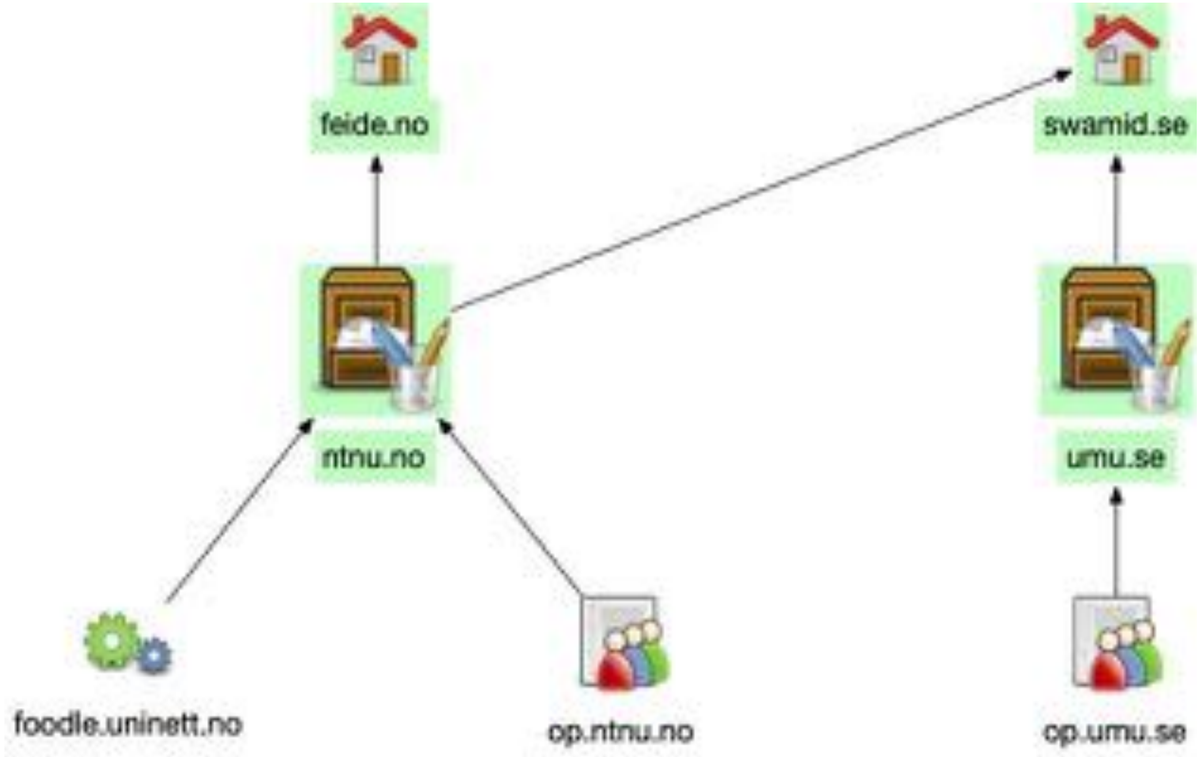
Entity Configuration

- All entities in a federation has a unique identifier (entity_id)
- All entities in a federation must publish information about itself.
- Use ‘Well known URI’, RFC 5785/8615 (.well-known/openid-federation) to construct publish URL.

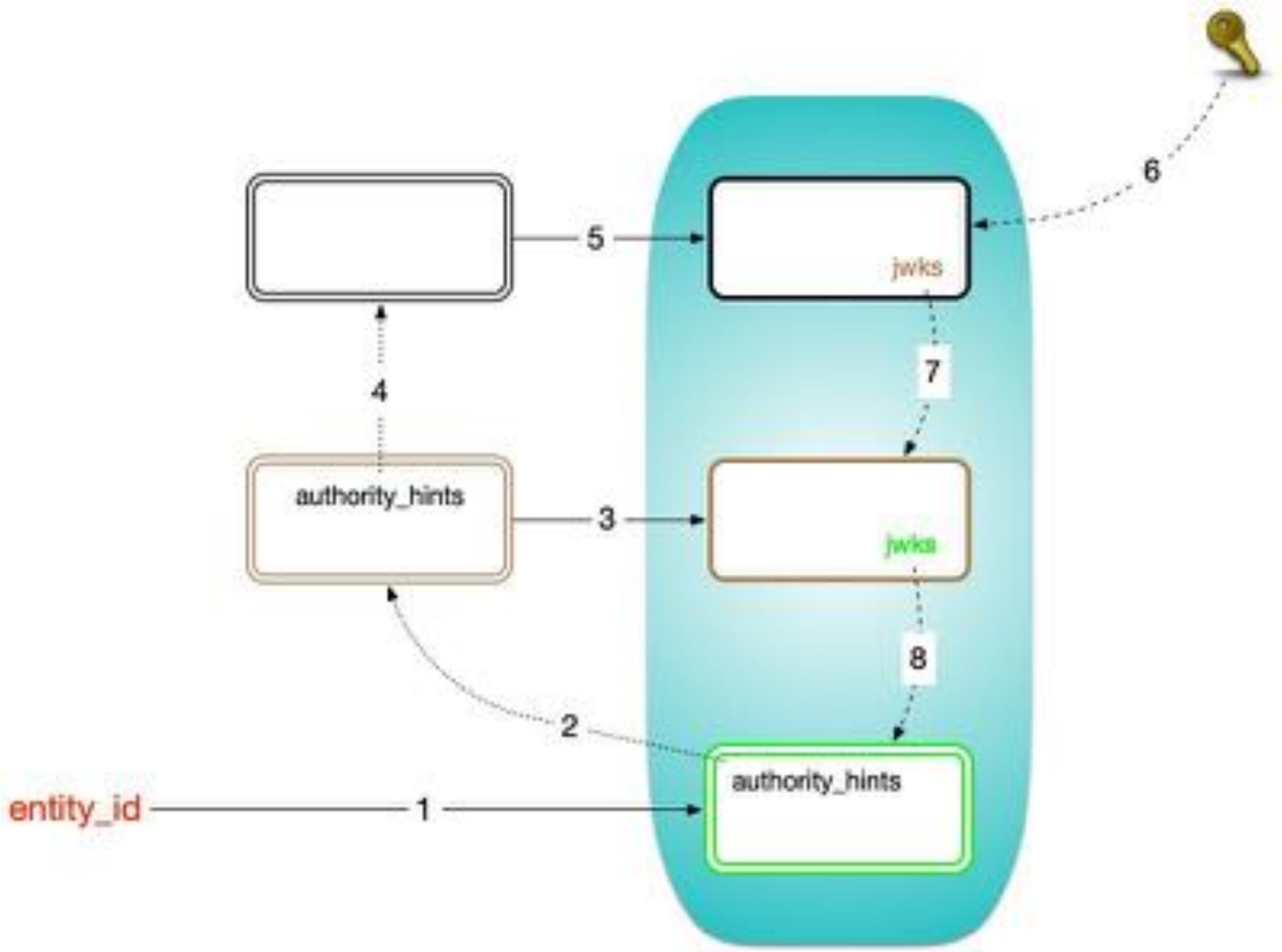


Fetch

- All entities that are expected to publish information about other entities must implement a fetch endpoint.
- All entities that has immediate subordinates are expected to publish information about those.



Collection of trust chain



swamid.se

umu.se

op.umu.se

Metadata policy

- add
- default
- one_of
- subset_of
- superset_of
- essential

```
{
  “metadata_policy”: {
    “openid_relying_party”: {
      “scopes”: {
        "subset_of": ["openid","eduperson"],
        "default": [“openid", "eduperson"]
      },
      "id_token_signed_response_alg": {
        "one_of": [“ES256", "ES384"],
        "default": "ES256"
      },
      "contacts": {
        "add": [
          "helpdesk@federation.example.org",
          "helpdesk@org.example.org"
        ]
      }
    }
  }
}
```

Trust Marks

Technically, trust marks as used by this specification are signed JWTs that represent a statement of conformance to a well-scoped set of trust and/or interoperability requirements.

Who can issue ’trust marks’

All entities in a federation !

- Trust Anchor/Federations operator
- Standardization unit
- Entity about itself - OIDF test suite
- Entity about another entity. AA about RP.

Trust mark introspection

- A trust mark issuer should provide a verification service. To which you can send a trust mark and get to know if it is still active.

Federation services

- Fetch
  - An authority about a subordinate -> entity statement
- Status
  - A trust mark issuer’s view on the status of a trust mark. -> True/False
- Resolve
  - A entity’s (the resolver) view of another entity -> {metadata, [trust_marks], [entity statement]}
- List
  - List of all subordinates to an entity -> [entity_id]

Client registration methods

- Automatic
  - The client does no registration! It just sends an authorization request with *client_id* == *entity_id* and client authentication method *private_key_jwt.*
  - The server must fetch and verify trust chains. Once it has the clients metadata it can verify the client authentication JWS with the clients keys.
- Explicit
  - The client does a dynamic registration. The registration request contains a self-signed entity statement.
  - The server fetches and verifies trust chains based on the self-signed entity statement.
  - The server responds with a entity statement describing its view of the client.