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

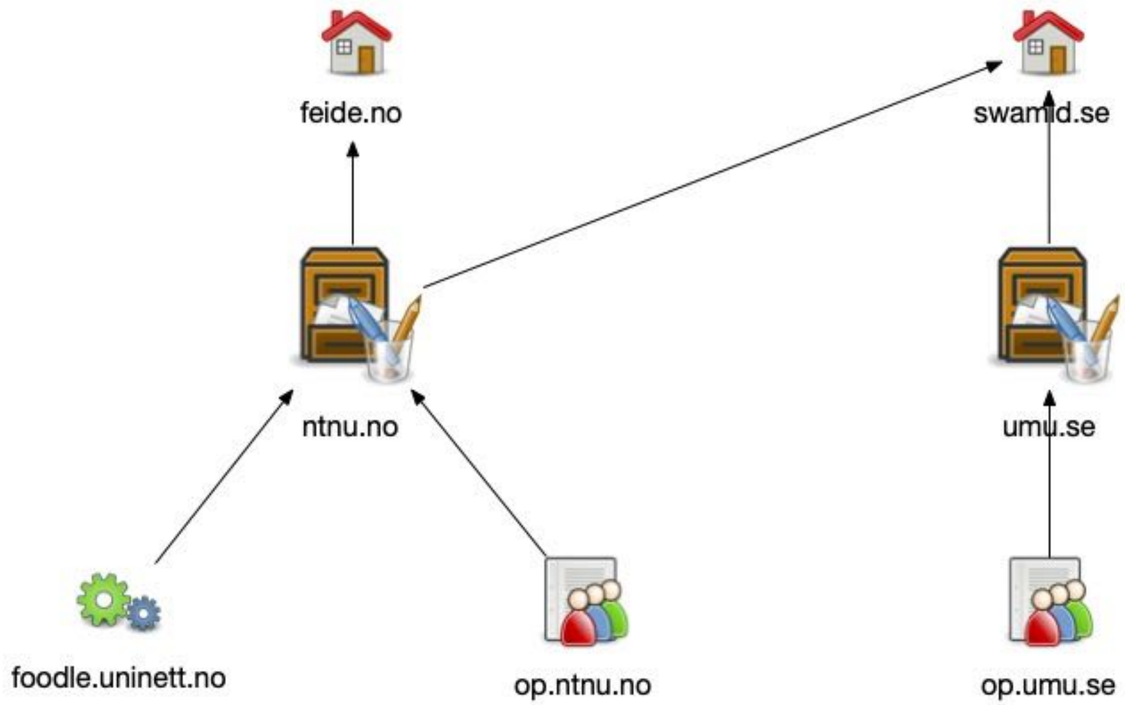
Entity Statement

- 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”
  ]
}
```

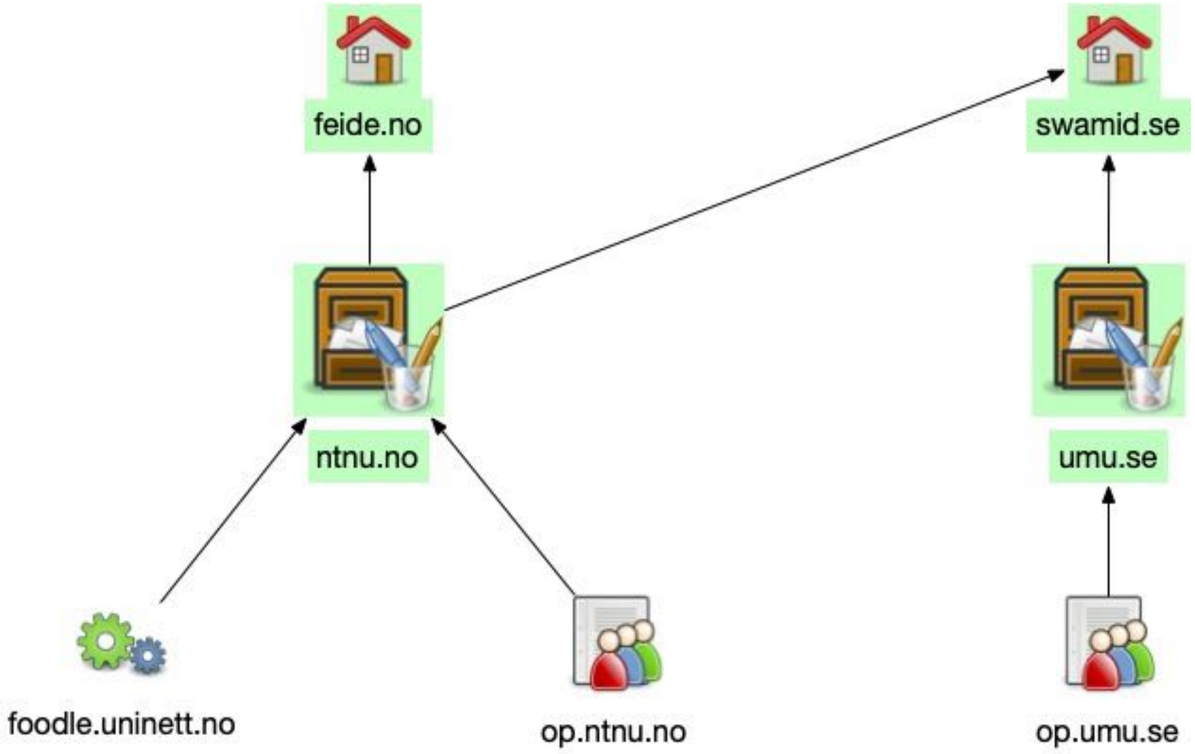
Entity Configuration

- 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.

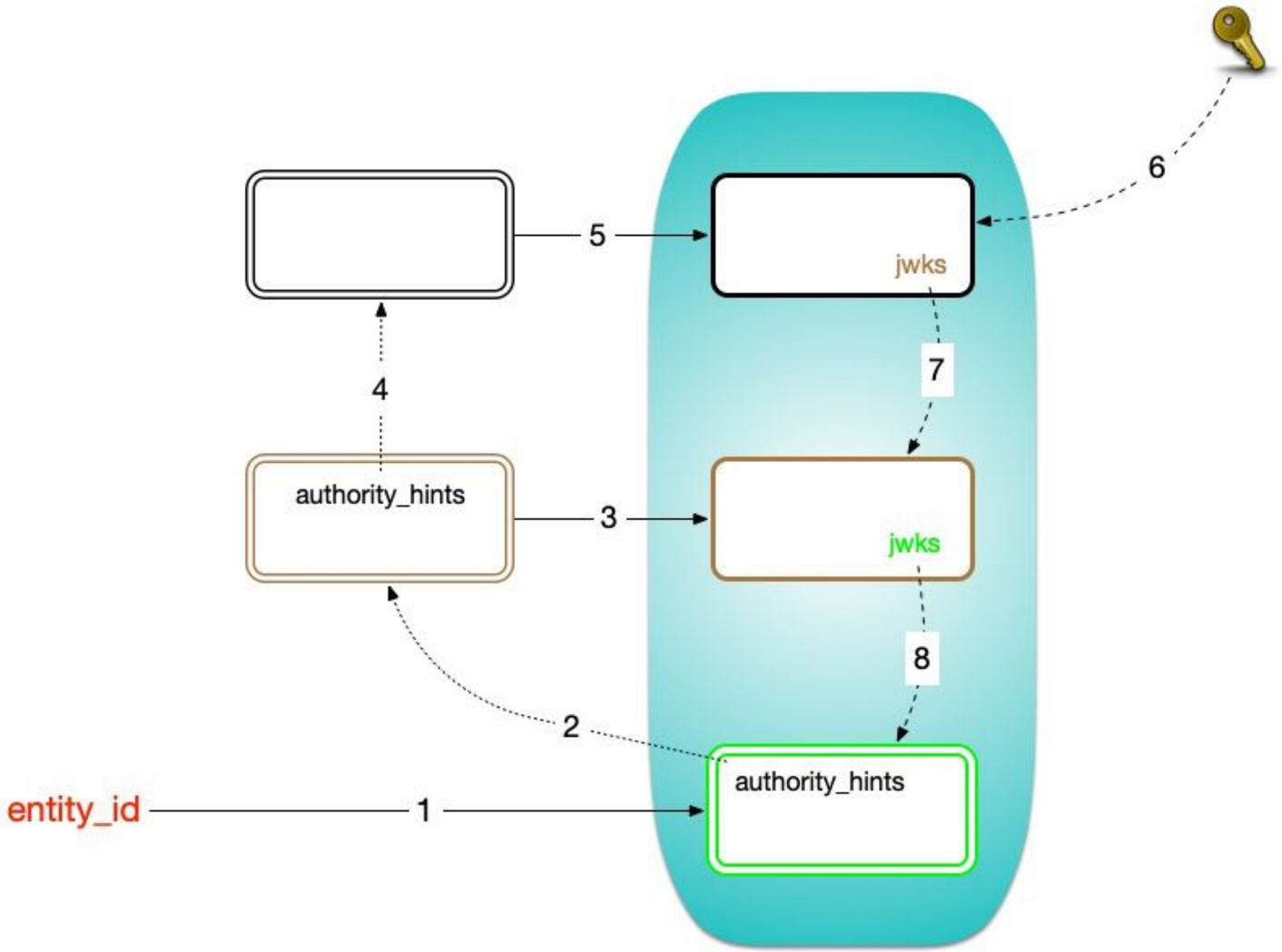


Fetch

- 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 a trust chain



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. Resource Server about an 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 for instance uses the 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.

Discover the Italian OIDC Federation

SPID/CIE the Italian eID systems

DIPARTIMENTO
PER LA TRASFORMAZIONE
DIGITALE

Giuseppe De Marco
Open Source Project Leader, Digital Identities
Department for Digital Transformation

MADE BY A TEAM

- **Roland Hedberg** (Author, independent)
- Agenzia per l'Italia Digitale (**AgID**)
- Istituto Poligrafico **(IPZS**)
- **Futuro e Conoscenza** (Bruno Kessler Found.)
- Department for Digital Transformation (Presidency of the Council of Ministers)

WITH

- **Automatic client registration**
- Adoption of **Trust Marks**
- Federation Intermediaries
- **Resolve Entity** endpoints
- Contributions to OIDC Federation 1.0 specs
- **Open Source SDKs** available on Developers Italia

Why OIDC Federation 1.0

- No longer need to fetch Metadata massively.
- RPs registration at OPs is completely dynamic.
- The Federation is transparent due to a realtime API.
- Federation intermediaries according to a Standard.
- Metadata of participants can change on the basis of policies adopted within the Federation, dynamically.

TWO NATIONAL DIGITAL IDENTITY SCHEMES, BOTH NOTIFIED ACCORDING TO THE EIDAS REGULATION

SPID

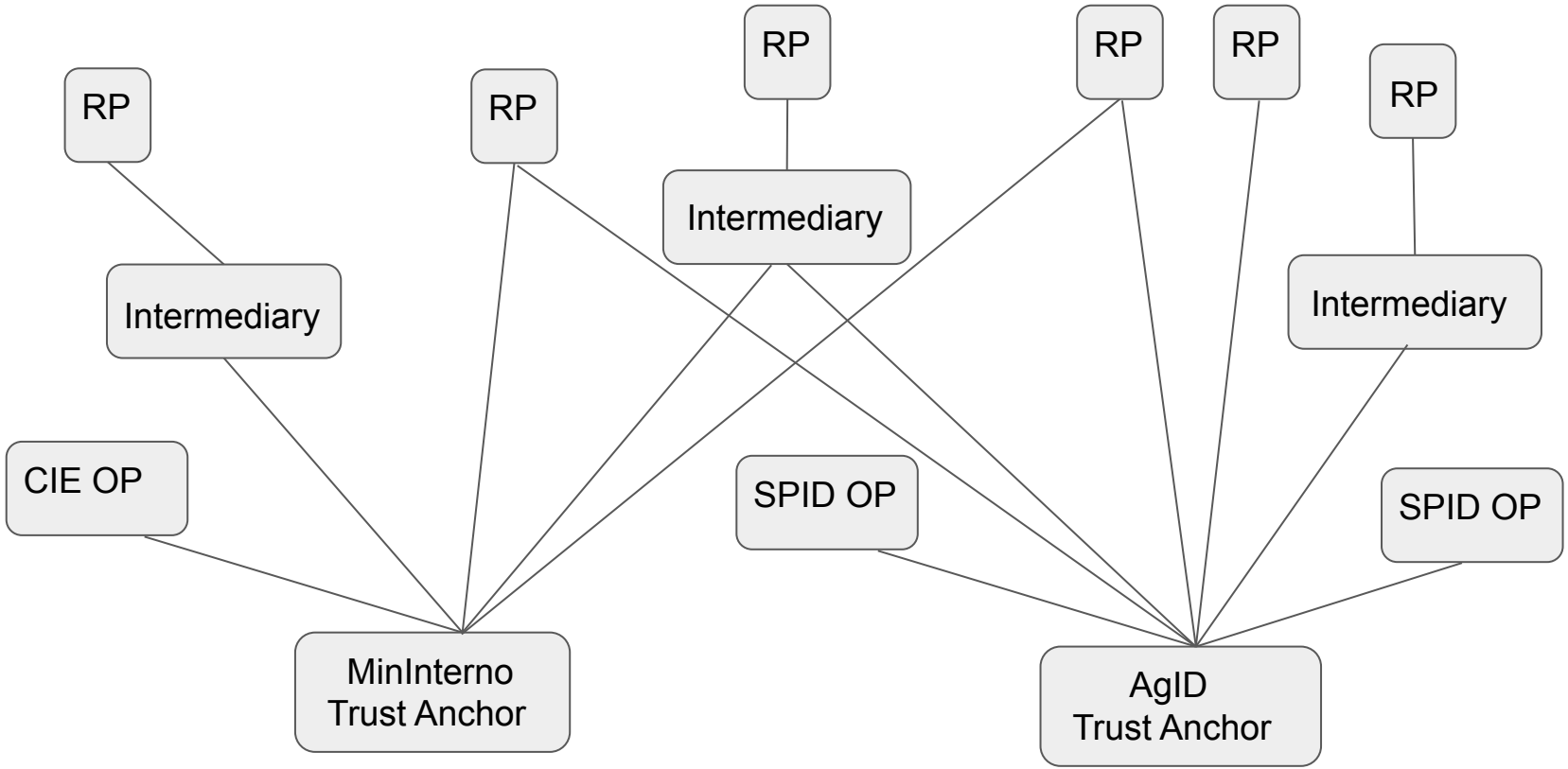
- **P**ublic **D**igital **I**dentity **S**ystem.
- Over **30 millions of identity** provided to citizens.
- Over **72 million accesses per month**.
- 9 Identity Providers.
- More than 12000 Service Providers.
- Born in 2017 with SAML2.
- Adopting OpenID Connect in 2022.
- Governance by AgID.

CIE id

- **E**lectronic **I**dentity **C**ard.
- Over **28 millions of identity** provided to citizens.
- Over **2 million accesses per month**.
- 1 Identity Provider.
- More than 7000 Service Providers.
- Born in 2019 with SAML2.
- Adopting OpenID Connect in 2022.
- Governance by Ministry of Interiors.
- IPZS is the supplier.

https://avanzamentodigitale.italia.it/it/progetto/spid

PORTRAIT OF THE ITALIANS EID SYSTEMS - SPID AND CIE COHEXIST THANKS TO OIDC FEDERATION



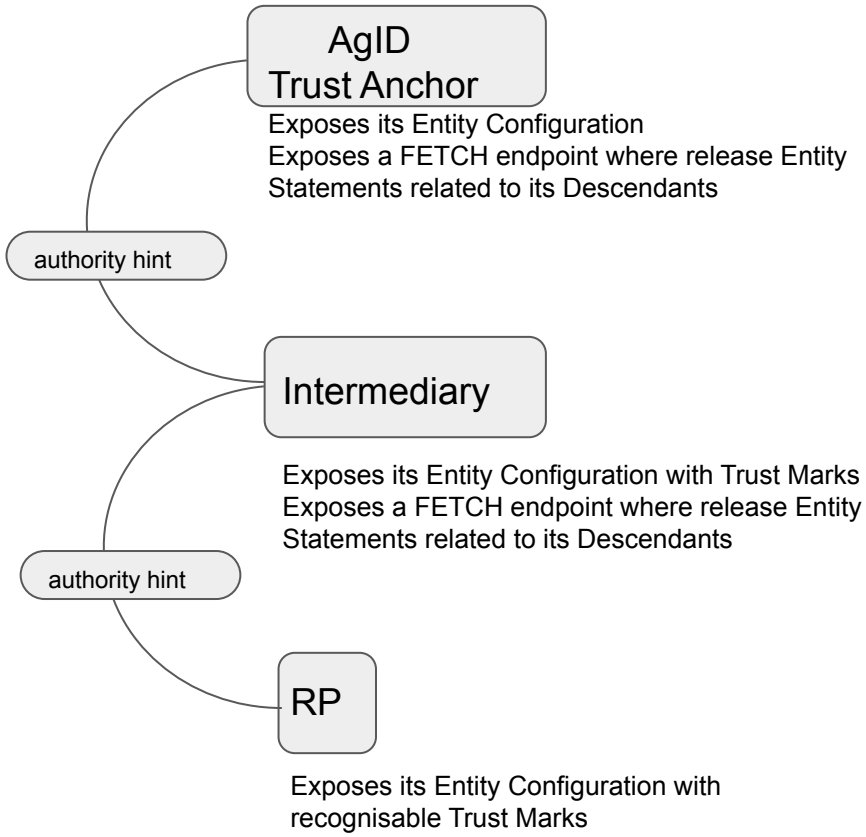
Two kinds of Intermediaries

LIGHT

- It handles descendants that supports OIDC Federation by their own.

FULL

- It acts like a IAM Proxy.
- It exposes the Entity Configurations of its descendants and the redirect_uris.



Trust Marks

- Signed by trusted and well known issuers.
- Provisioned to participants when they join in the CIE or SPID Federation.
- In absence of at least one recognisable and valid Trust Mark, the Metadata Discovery does not take place on the verifier side*

Automatic client registration

The OP:

1. Has Trust Anchor's public keys and Entity Configuration.
2. Gets the **trust_mark_issuers** and the **max_path_length**.
3. Fetches the **client_id** from the incoming authorization request. Eg: **https://rp.example.it/spid/**.
4. Fetches the Entity Configuration at **https://rp.example.it/spid/.well-known/openid-federation**.
5. Checks if it contains a valid **Trust Mark** for a RP profile. If false, the request is ignored and the *client_id* may be temporary banned.
6. Follows the path given by **authority hints** to build and validate the Trust Chain.
7. Builds the final metadata, stores it and keeps it until it expires, then updates it periodically (24h)*.

HOW A RELYING PARTY BUILDS THE SPID BUTTON

Fetches all the leafs of type Openid Provider from the Federation listing endpoint of the Trust Anchor

Fetches the Entity Configuration for each of the OPs

Validates the Entity Configurations of the OPs with the Entity Statements released by Trust Anchor

Apply the Federation Policy and produces the final metadata

From each metadata extracts

- Logo uri
- Organization Name
- subject url (entity id)

A CUSTOMIZATION FOR THE FEDERATION LIST ENDPOINT

Federation List Endpoint lists a JSON object containing all the descendants of a federation entity.

In addition of the official specifications we added the parameter **entity_type**

```
GET /list?entity_type=openid_provider
HTTP/1.1
Host: registry.agid.gov.it

200 OK
Last-Modified: Wed, 22 Jul 2018 19:15:56 GMT
Content-Type: application/json

["https://openid-provider.it/", … ]
```

picture by keblog.it / Francois Nielly

WATCH THE FEDERATION FROM THE EYES OF A PARTICIPANT

Resolve Entity endpoint

- Enables the possibility to check the metadata freshness.
- Gives the cached trust chains.
- Won’t trigger a new Metadata Discovery on each HTTP Request.
- Carries in the response the proof of the trust chain calculated, as array of the original entity statements.

Each participant MUST deploy a Resolve endpoint.

photo by https://www.policymed.com/

THE COMMUNITY MODEL BEHIND THE METHOD

Contributions to the specs

Participating in the work of the OpenID Foundation's WG Connect A/B as independent individuals

Proposing ideas and offering contributions through https://bitbucket.org/openid/connect/

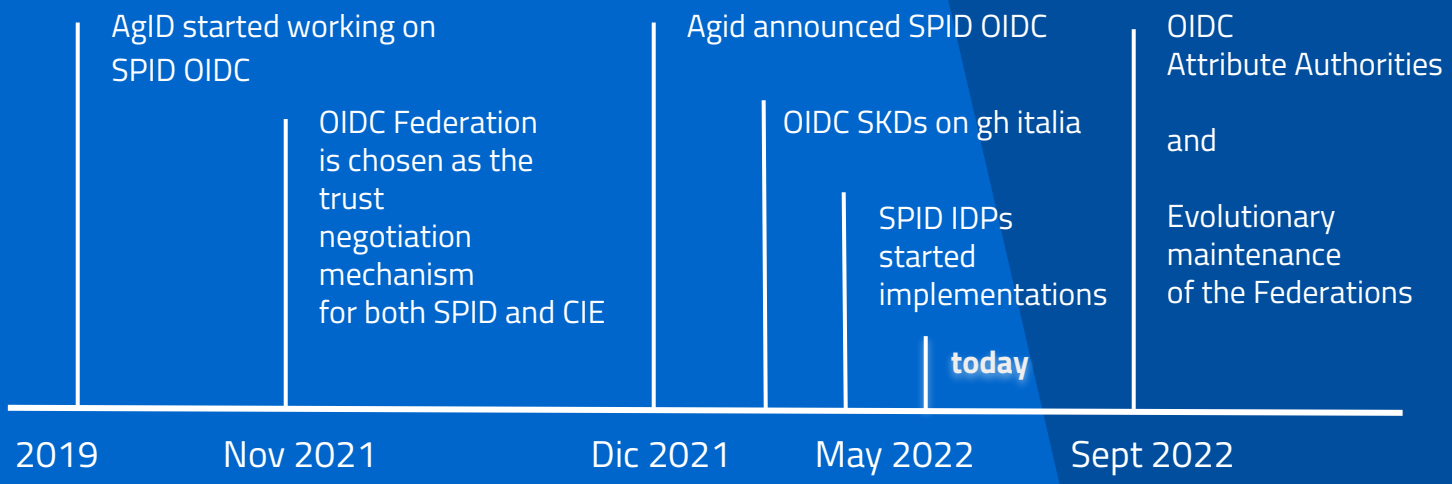
Federation API has been divided into specialized endpoints.
Trust Marks revamped with non normative examples.
Trust Mark Status endpoint.
Entity Resolve endpoint.
Content types for Trust Marks and Entity Statements.

Several implementations in different programming languages.
CI and **Docker images** included!

- Python
- PHP
- Java
- C# / ASPnet Core
- NodeJS

https://github.com/orgs/italia/repositories?q=spid-cie-oidc

The Italian OpenID Connect timeline



DIPARTIMENTO
PER LA TRASFORMAZIONE
DIGITALE

Thank you

https://developers.italia.it/

@InnovazioneGov

@DipartimentoTrasformazioneDigitale

@company / ministeroinnovazione/

DIPARTIMENTO
PER LA TRASFORMAZIONE
DIGITALE