OIDC Federations Spec Overview

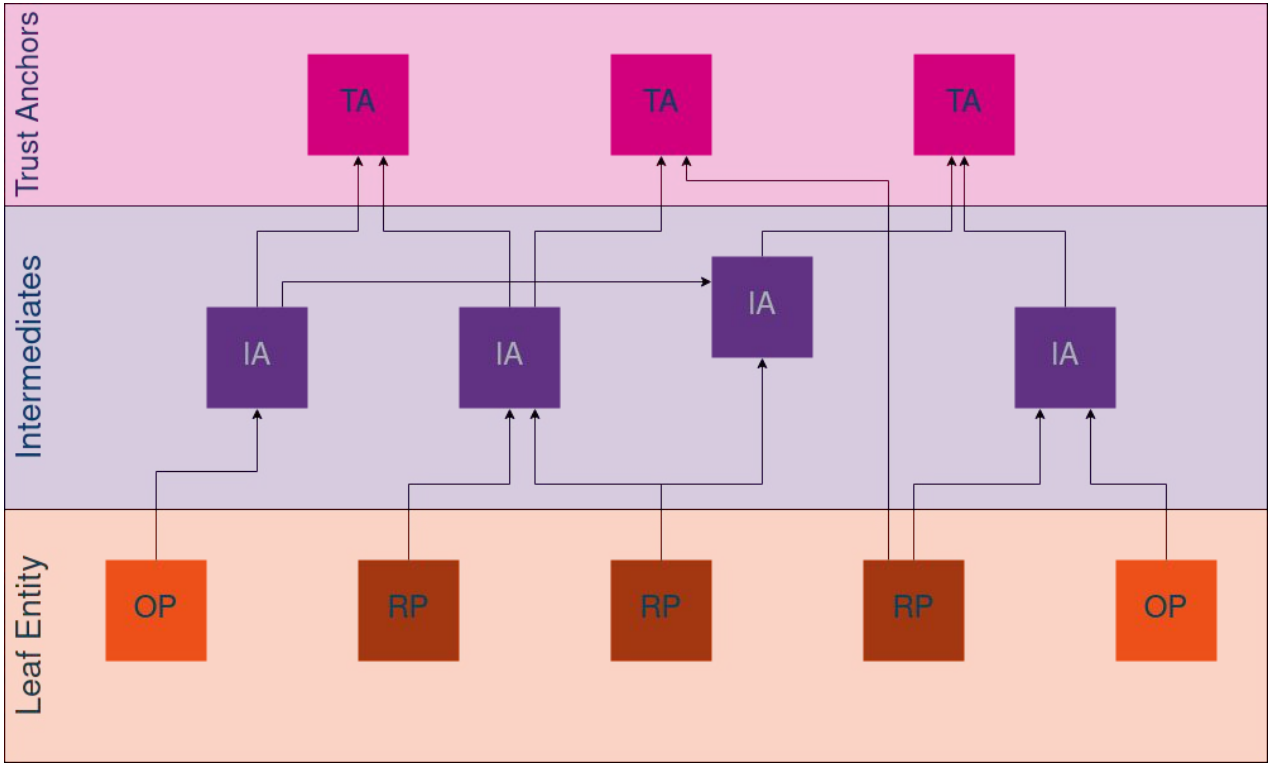
Gabriel Zachmann, KIT

~~2023-02-21~~

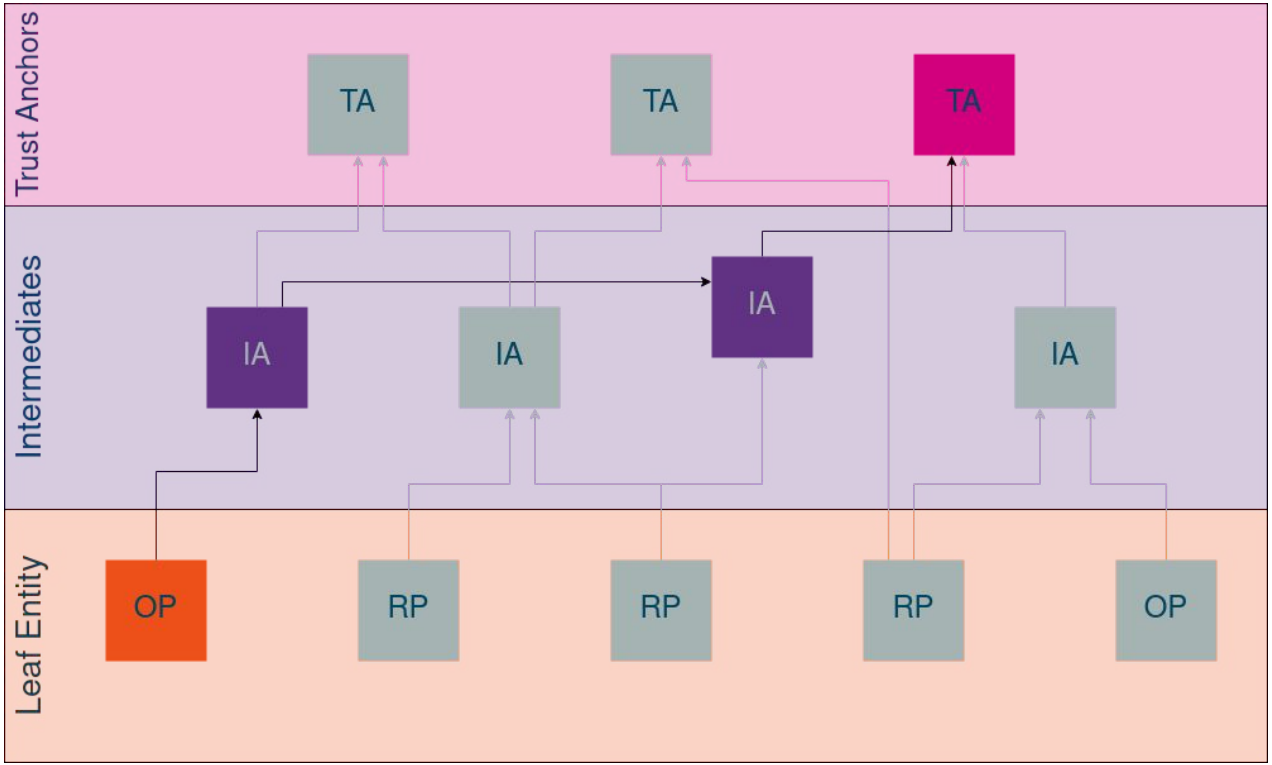
~~2023-02-28~~

2023-03-07

- OIDC Federation Specification
- Authlete (Comm Java Impl) Docu with graphs


Trust Anchors
Intermediates
Leaf Entity
TA
TA
TA
IA
IA
IA
IA
OP
RP
RP
RP
OP

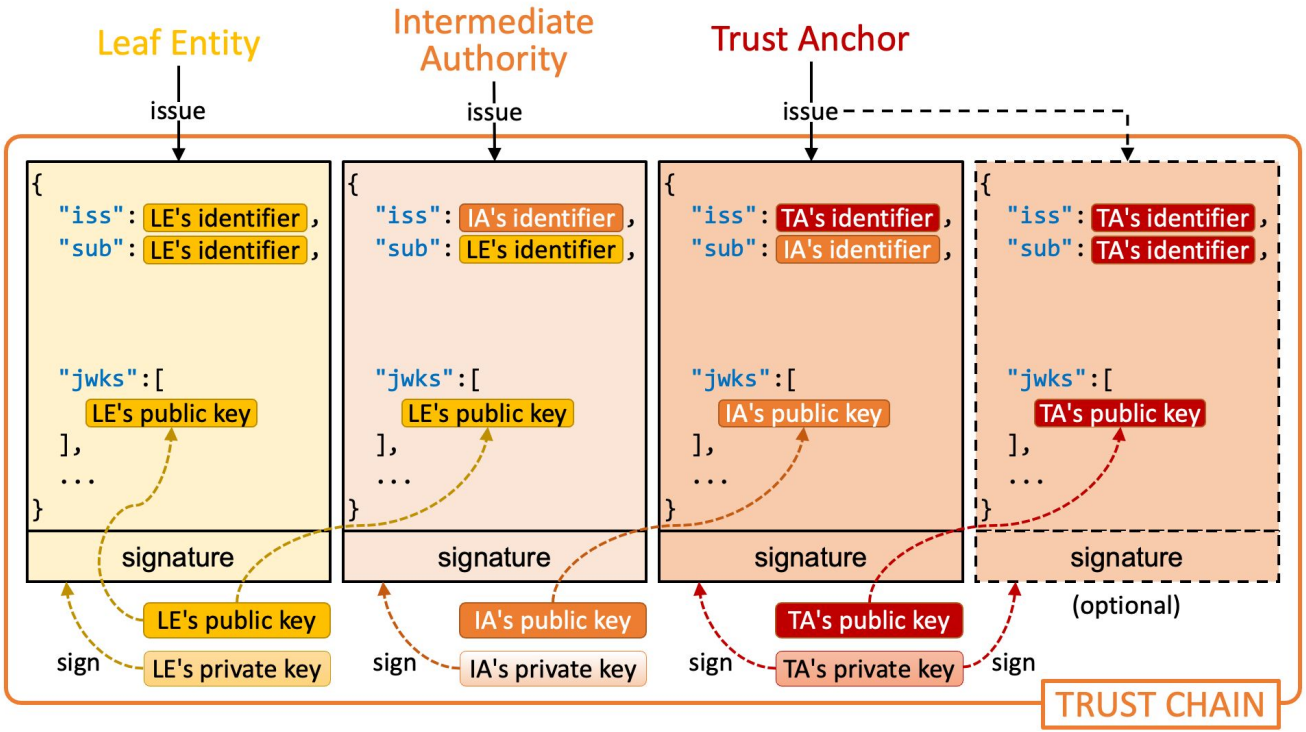
- Entity Identifier
  - URI -> OP: issuer url
- Entity Statement
  - Signed JWT
  - Contains information needed for the subject's entity to participate in federation
- Entity Configuration:
  - Self-signed Entity Statement
  - Endpoint: /.well-known/openid-federation
- Trust Chain
  - Chain of Entity Statements from a Leaf Entity [via Intermediates] to a Trust Anchor


Trust Anchors
TA
TA
TA
Intermediates
IA
IA
IA
IA
Leaf Entity
OP
RP
RP
RP
OP

- Entity Identifier
  - URI -> OP: issuer url
- Entity Statement
  - Signed JWT
  - Contains information needed for the subject's entity to participate in federation
- Entity Configuration:
  - Self-signed Entity Statement
  - Endpoint: /.well-known/openid-federation
- Trust Chain
  - Chain of Entity Statements from a Leaf Entity [via Intermediates] to a Trust Anchor
- OIDC Metadata
  - Combined from a Trust Chain
- Federation Endpoint

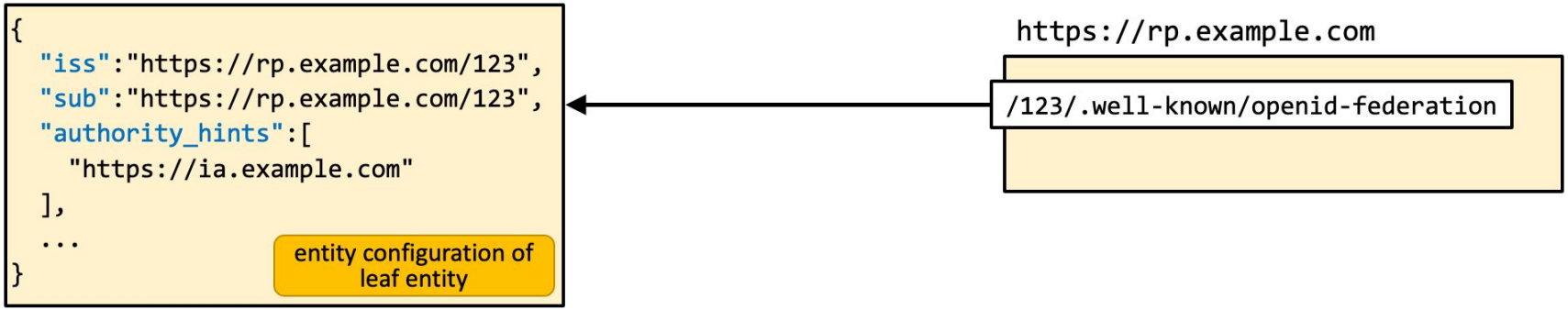
Trust Chain

Trust Chain


Leaf Entity
Intermediate Authority
Trust Anchor
issue
issue
issue
{
"iss": LE's identifier,
"sub": LE's identifier,
"jwks":[
LE's public key
],
...
}
{
"iss": IA's identifier,
"sub": LE's identifier,
"jwks":[
LE's public key
],
...
}
{
"iss": TA's identifier,
"sub": IA's identifier,
"jwks":[
IA's public key
],
...
}
{
"iss": TA's identifier,
"sub": TA's identifier,
"jwks":[
TA's public key
],
...
}
signature
signature
signature
signature
(optional)
LE's public key
IA's public key
TA's public key
sign
LE's private key
sign
IA's private key
sign
TA's private key
sign
TRUST CHAIN

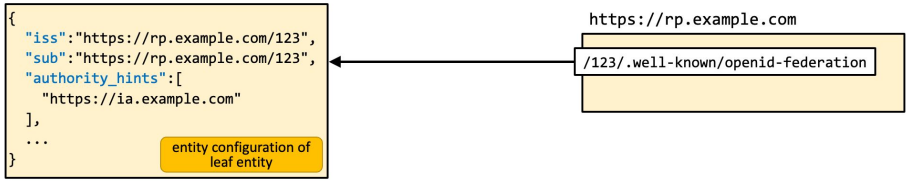
Trust Chain Resolution

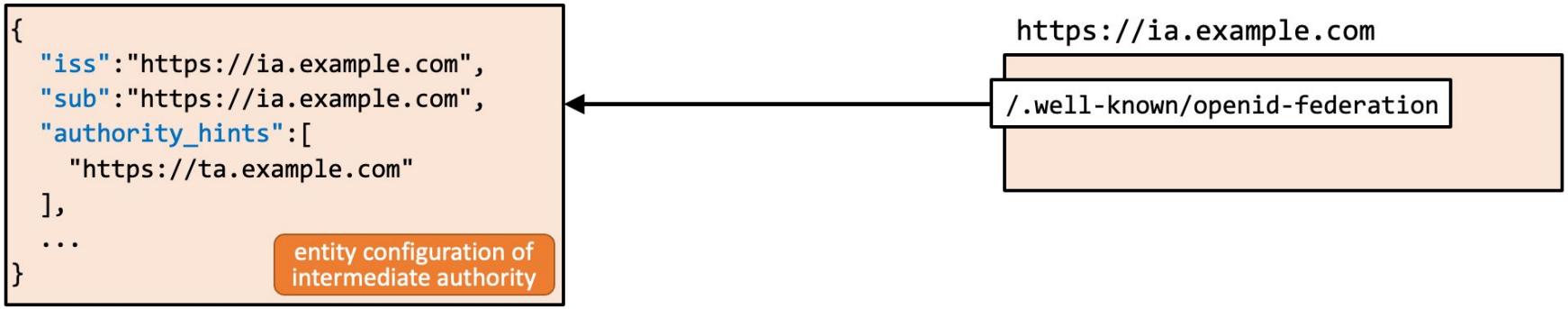
1. Getting self-signed Entity Configuration -> /.well-known/openid-federation



Trust Chain Resolution

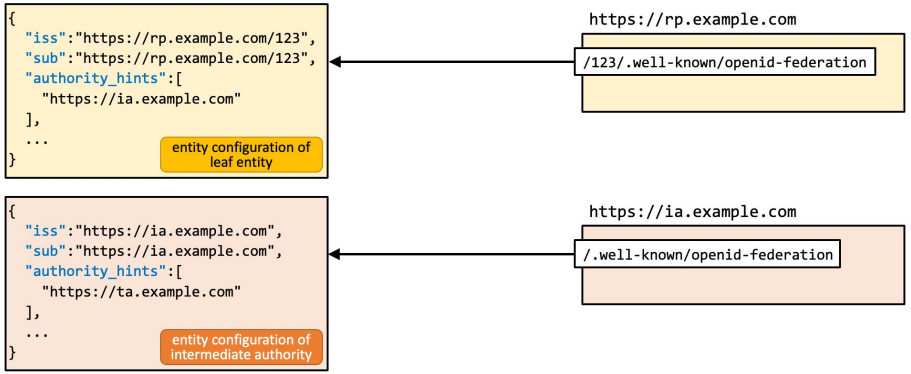
1. Getting self-signed Entity Configuration
2. Read authority_hints
3. Iterate on the possible Intermediates:
   a. Obtain Entity Configuration
   b. Read authority_hints
      i. Found: Goto 3.

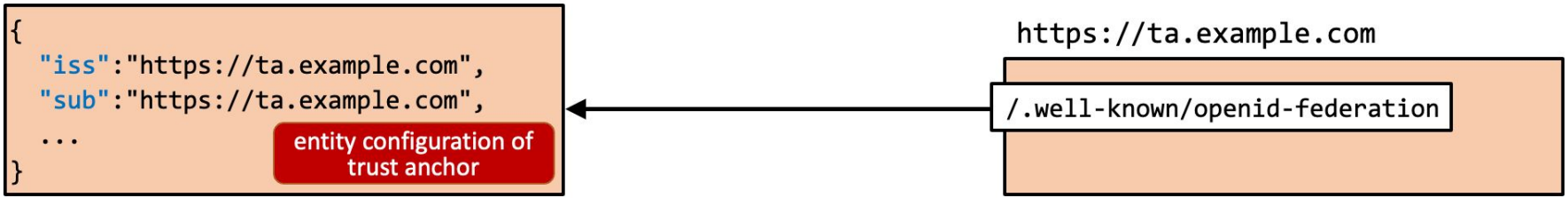




Trust Chain Resolution

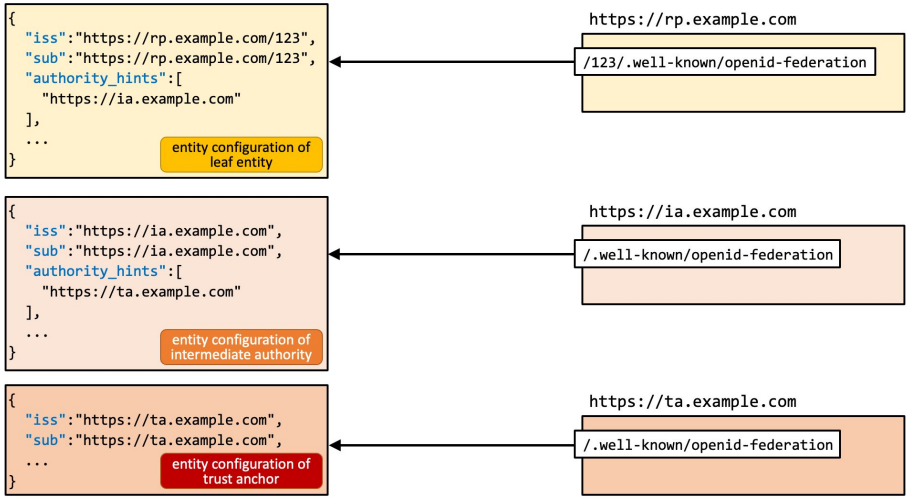
1. Getting self-signed Entity Configuration
2. Read authority_hints
3. Iterate on the possible Intermediates:
   a. Obtain Entity Configuration
   b. Read authority_hints
      i. Found: Goto 3.
      ii. None: Found a (possible) TA
         1. Trusted: Good
         2. Not Trusted: Ignore





Trust Chain Resolution

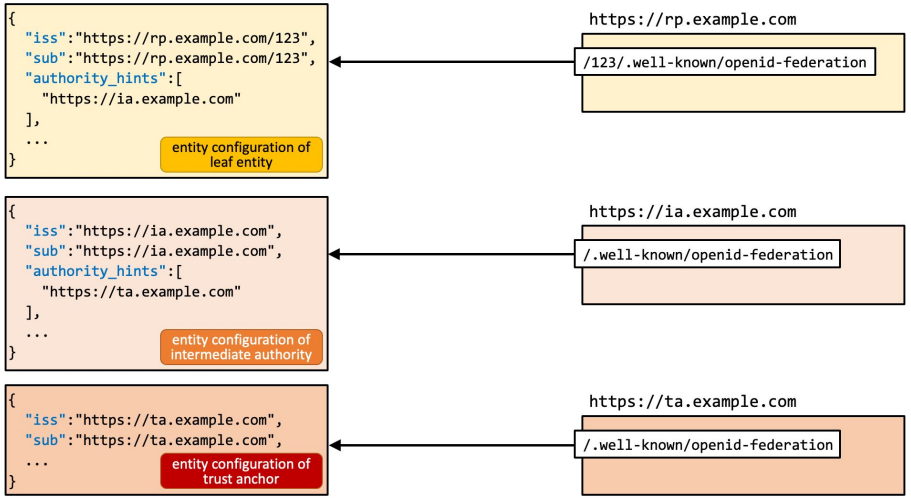
1. Getting self-signed Entity Configuration
2. Read authority_hints
3. Iterate on the possible Intermediates:
   a. Obtain Entity Configuration
   b. Read authority_hints
      i. Found: Goto 3.
      ii. None: Found a (possible) TA
         1. Trusted: Good
         2. Not Trusted: Ignore



We now have one or more chains from the Leaf Entity to a Trust Anchor.
But we did not check the way down.

Trust Chain Resolution

1. Getting self-signed Entity Configuration
2. Read authority_hints
3. Iterate on the possible Intermediates:
   a. Obtain Entity Configuration
   b. Read authority_hints
      i. Found: Goto 3.
      ii. None: Found a (possible) TA
         1. Trusted: Good
         2. Not Trusted: Ignore



4. On IA and TA fetch Entity Statements about their subordinate from their Federation Fetch Endpoint.

Trust Chain Resolution

1. Getting self-signed Entity Configuration -> /.well-known/openid-federation
2. Read authority_hints
3. Iterate on the possible Intermediates:
    a. Obtain Entity Configuration
    b. Read authority_hints
        i. Found: Goto 3.
        ii. None: Found a (possible) TA
            1. Trusted: Good
            2. Not Trusted: Ignore
4. On IA and TA fetch Entity Statements about their subordinate from their Federation Fetch Endpoint.
5. Verify Chain: Chaining, Signatures, Expiration
6. Choose one chain
7. Calculate expiration time of chain

Trust Chain Resolution - Flow

Trust Chain Resolution

Trust Chain Resolution - Flow

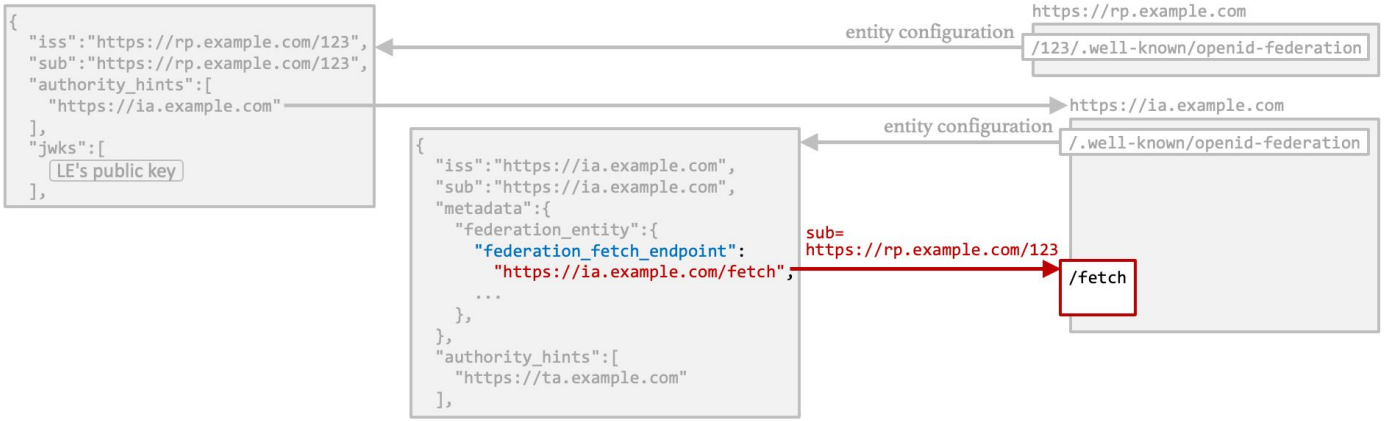
Trust Chain Resolution

Trust Chain Resolution - Flow

Trust Chain Resolution

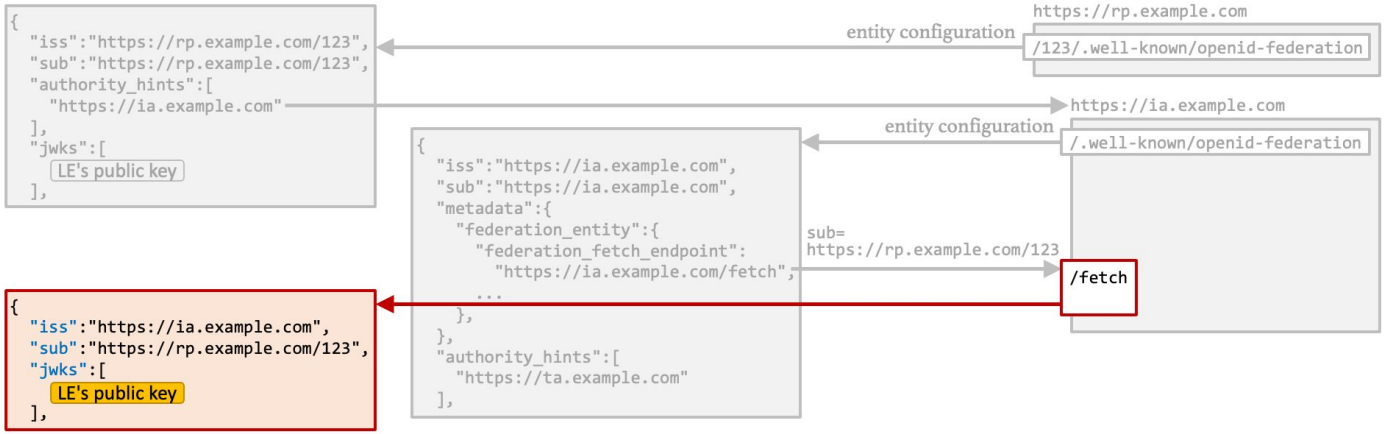
Trust Chain Resolution - Flow

Trust Chain Resolution

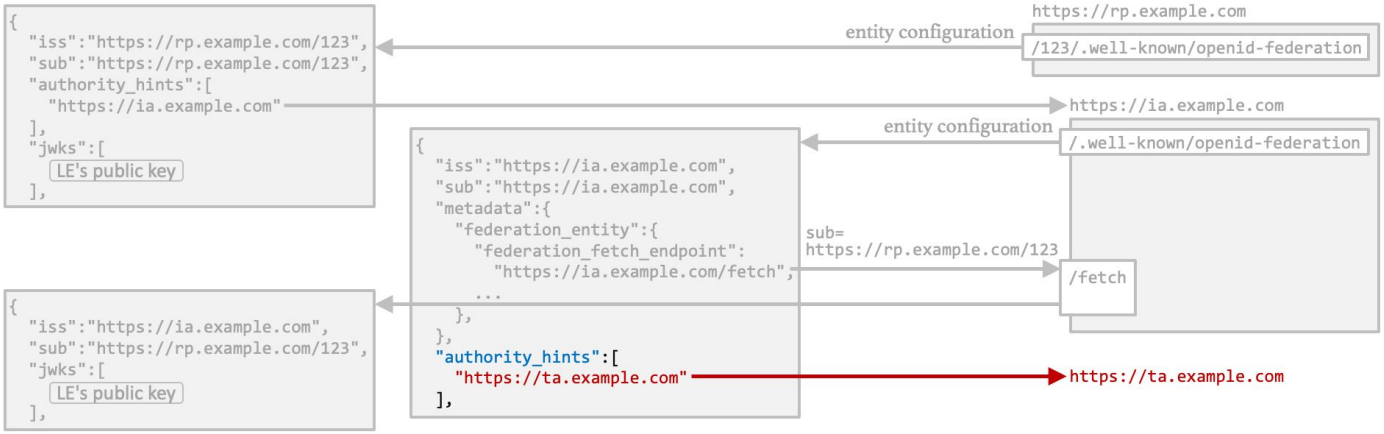


Trust Chain Resolution - Flow

Trust Chain Resolution

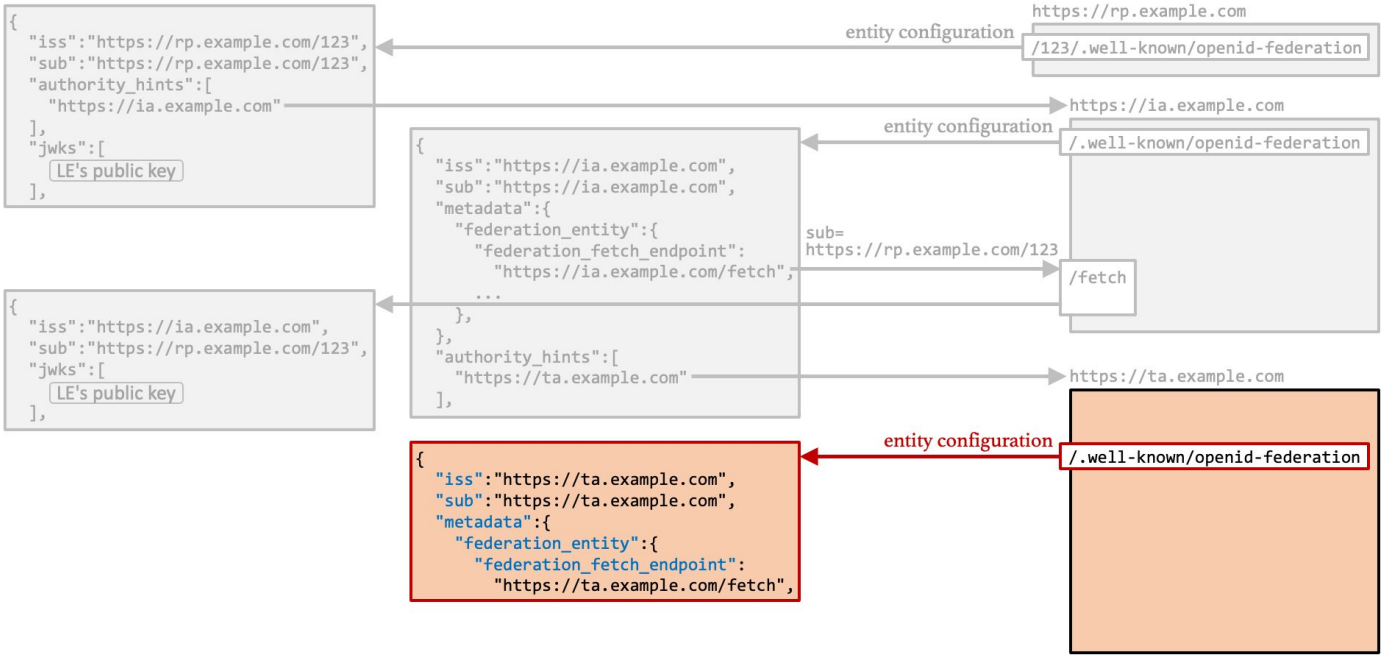


Trust Chain Resolution



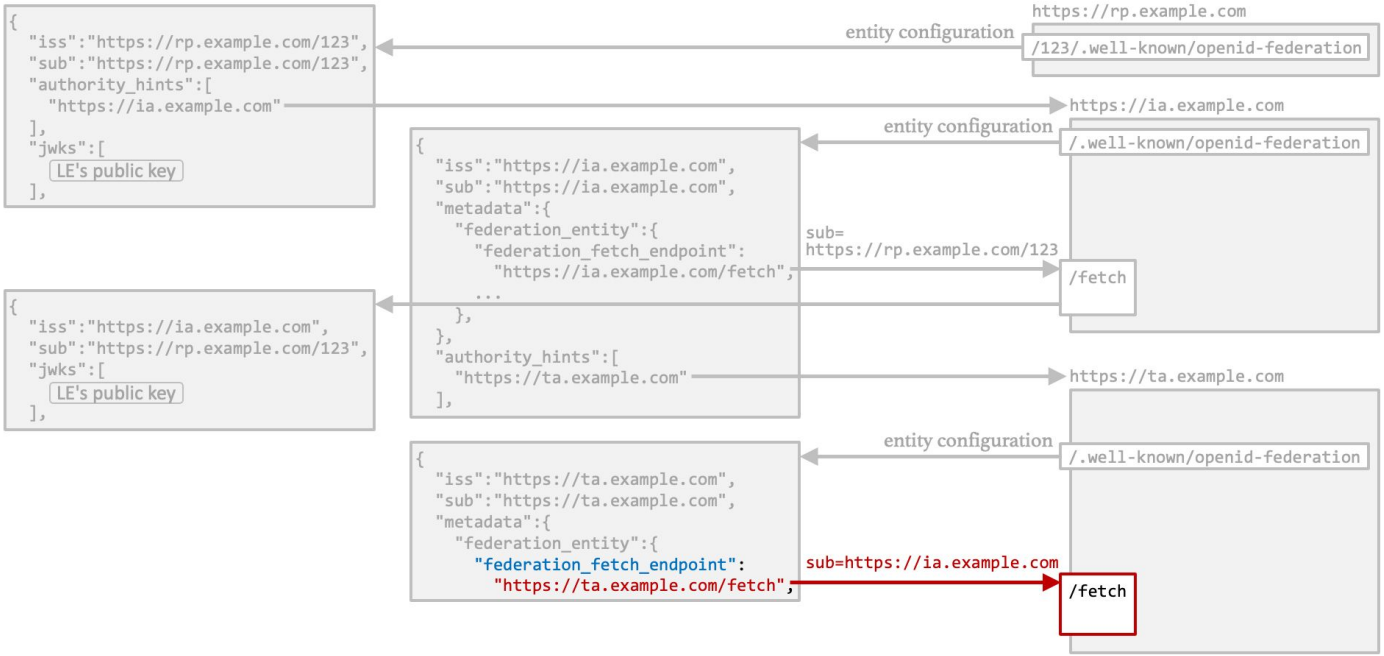
Trust Chain Resolution - Flow

Trust Chain Resolution



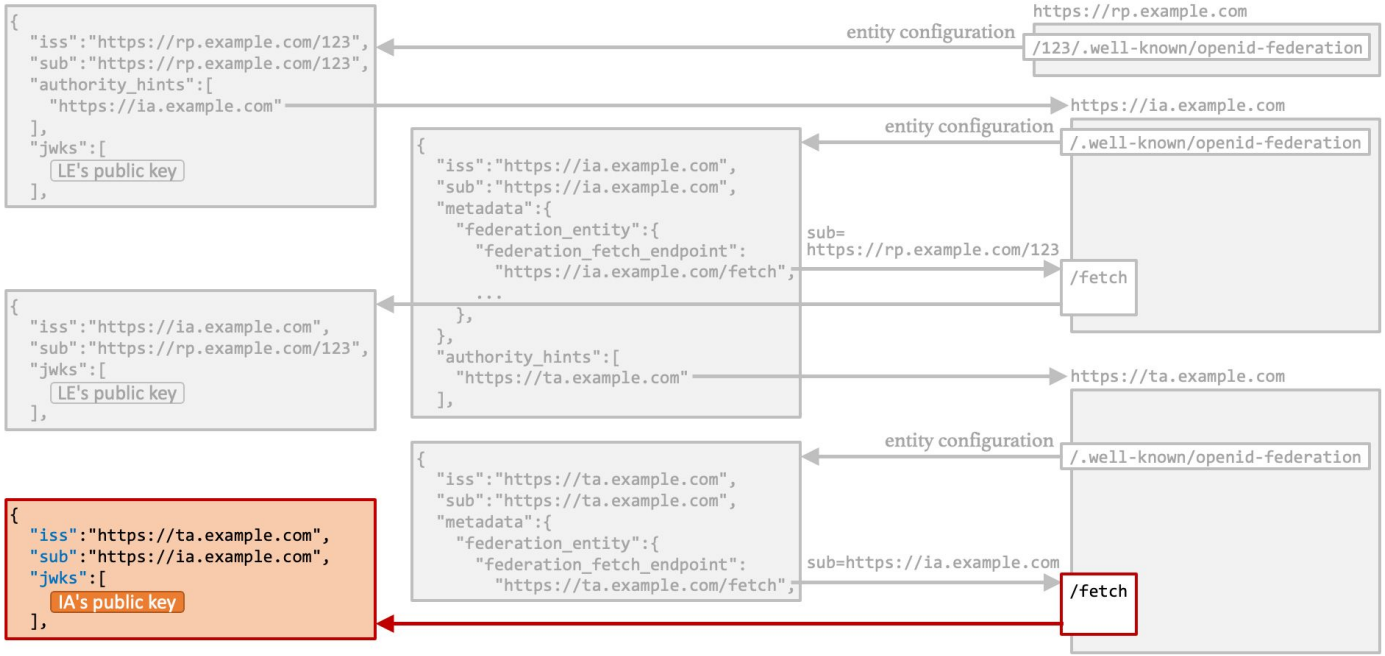
Trust Chain Resolution - Flow

Trust Chain Resolution



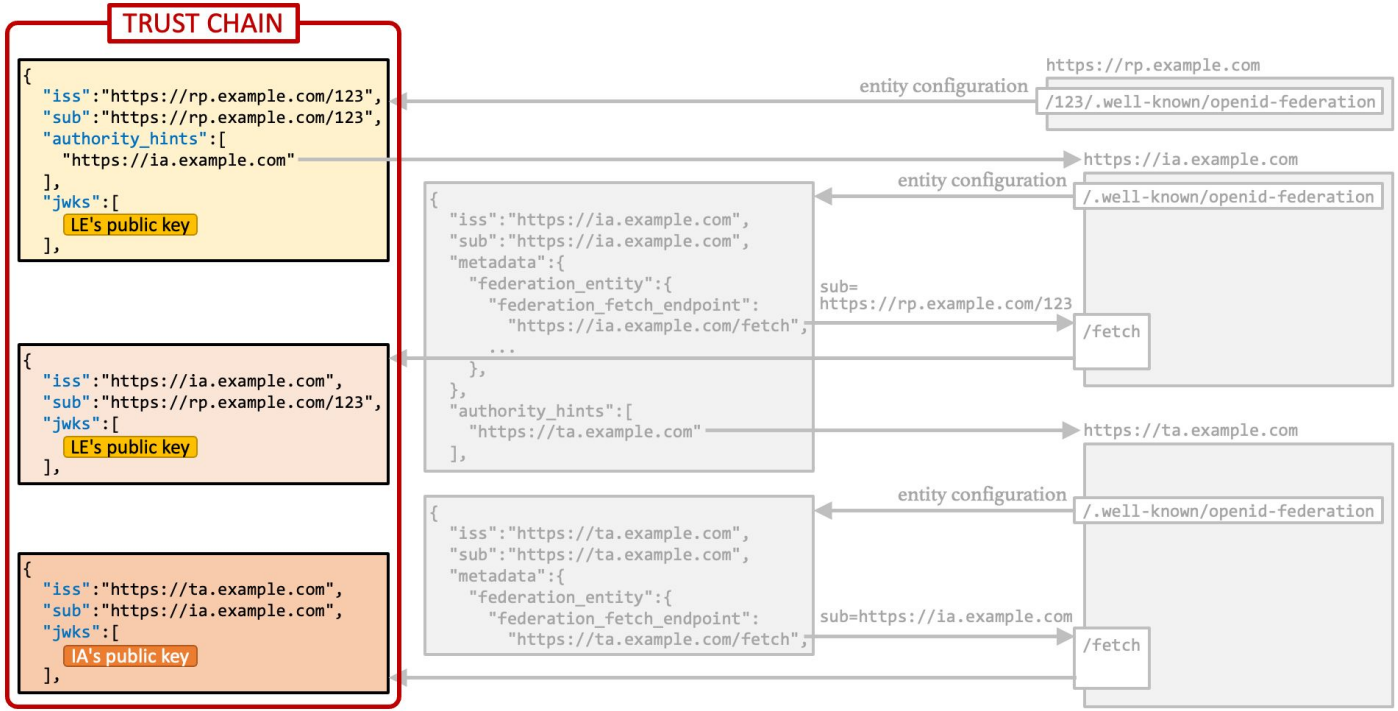
Trust Chain Resolution - Flow

Trust Chain Resolution



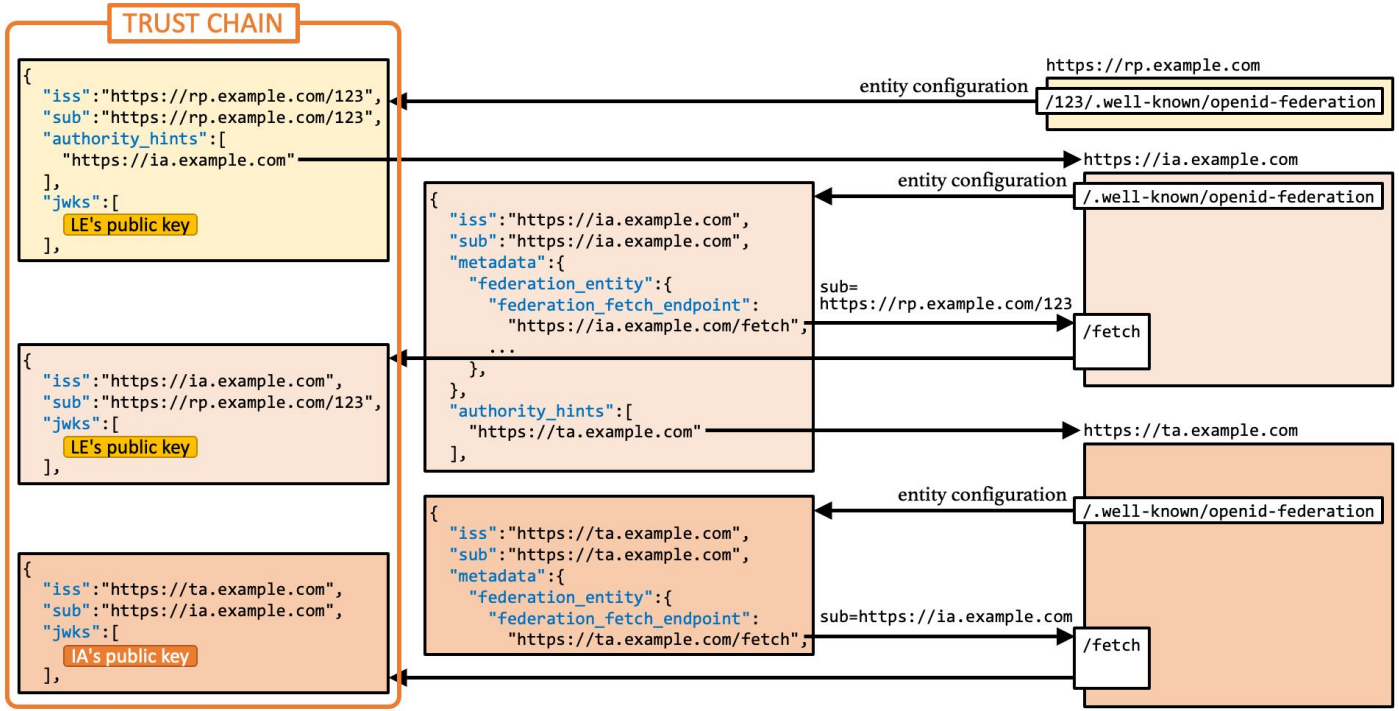
Trust Chain Resolution - Flow

Trust Chain Resolution



Trust Chain Resolution - Flow

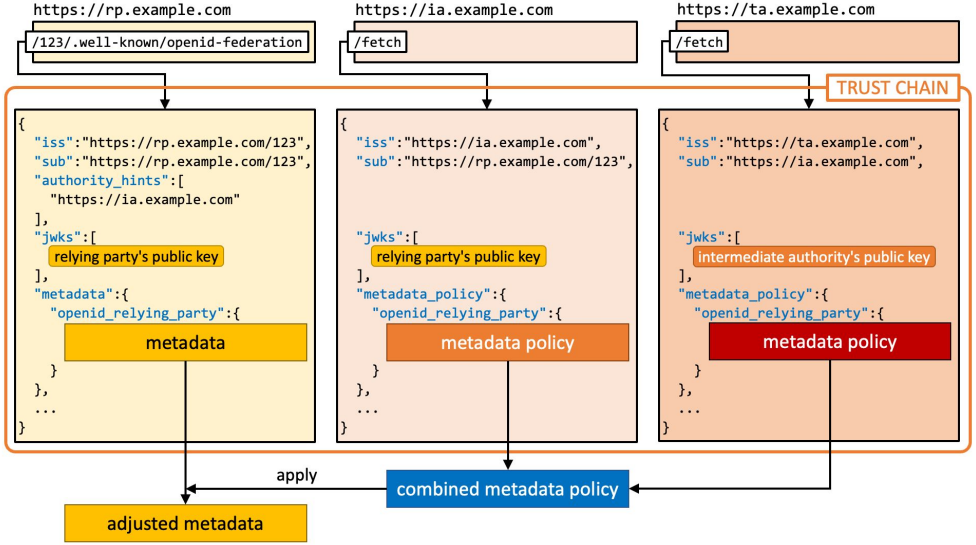
Trust Chain Resolution



Metadata

Metadata

- A Leaf Entity's metadata is obtained by:
  - Combining metadata_policy in the chain
  - Applying the policy to the metadata in its Entity Configuration



Metadata

- Different metadata types:
  - openid_relying_party oauth_client oauth_resource
    - client metadata + client_registration_types
  - openid_provider oauth_authorization_server
    - OP metadata + metadata about registration + auth
  - federation_entity
    - endpoints, fed metadata
- General claims for all types:
  - organization_name, signed_jwks_uri, jwks

Metadata Policy

- metadata_policy has **policy entries**:
  - Metadata parameter, e.g. id_token_signed_response_alg
  - One or more **operators**: *value modifiers* or *value checks*
- Example:

```
"metadata_policy" : {
  "openid_relying_party": {
    "id_token_signed_response_alg": {
      "default": "ES256",
      "one_of" : ["ES256", "ES384", "ES512"]
    }
  }
}
```

Metadata Policy - Operators

- Value modifiers:
  - value — Set this value
  - add — Add this value (if not present)
  - default — Set this value if none set
- Value checks:
  - essential — Indicates if a value is required
  - one_of — Value must be one of the listed
  - subset_of — Intersection
  - superset_of — Defines values must be included

Federation Endpoint

Federation Endpoint

- **Fetch Endpoint**
- **List Endpoint**
- **Resolve Endpoint**
- **Trust Mark Status Endpoint**
- Historical Keys Endpoint
  - /.well-known/openid-federation-historical-jwks

Federation Fetch Endpoint

- Used to collect entity statements when building the trust chain.

- Example:
  - GET https://ia.example.com/fetch?sub=https://rp.example.com/123
- Response: Entity Statement

Federation Entity Listing

- Query list of all Entities immediately subordinate
- Can be filtered by entity_type

Example response:

```
200 OK
Content-Type: application/json

[
  "https://ntnu.andreas.labs.uninett.no/",
  "https://blackboard.ntnu.no/openid/callback",
  "https://serviceprovider.andreas.labs.uninett.no
      /application17"
]
```

Federation Resolve Endpoint

- Fetch resolved metadata as trusted by the resolver
- GET Request: sub, anchor, [type]
- Response: **metadata**, trust_marks, trust_chain

Trust Mark Status

- Used to check whether a Trust Mark is still active or not.
- Query is sent to the trust mark issuer.

Trust Marks

Trust Marks?

- Signed JWT -> Signed by a federation-accredited authority
- Content: iss, sub, id, iat, logo_uri, exp, ref

```
{
  "iss": "https://secusign.org",
  "sub": "https://example.com/op",
  "iat": 1579621160,
  "id": "https://secusign.org/level/A",
  "logo_uri": "https://secusign.org/static/levels/
    certification-level-A-150dpi-90mm.svg",
  "ref": "https://secusign.org/conformances/"
}
```

- Trust Anchors can restrict who can issue a certain trust mark

```
"trust_marks_issuers": {
  "https://openid.net/certification/op": ["*"],
  "https://refeds.org/wp-content/uploads/2016/01/Sirtfi-1.0.pdf":
    ["https://swamid.se"]
}
```

OIDC Communication

OIDC Communication

- No prior registration between RP and OP
- How is trust & configuration established?

Two types:

- Automatic Registration
- Explicit Registration

Automatic Registration

- RP does no Registration!
- RP uses Entity Identifier as the client_id
- OP fetches and verifies trust chains. -> client metadata
  - Verify request authentication

OP can decide if a Auth request uses OIDC fed automatic registration:

- OP supports OIDC Fed and Automatic Registration
- Incoming client_id is a valid URL
- Client ID is not a registered client

Explicit Registration

- RP does explicit registration with OP prior to other requests
- Similar to Dynamic Client Registration, but with Trust Chains
- Federation Registration Endpoint

Explicit Registration - RP View

1. Obtain list of acceptable Trust Anchors of the OP
2. Choose a subset to progress with
3. Filter authority_hints to only the ones that have a route to the selected TAs
4. Construct Entity Configuration with metadata influenced by OP's metadata
5. Optionally: Construct Trust Chain
6. POST Entity Configuration / Trust Chain in the request body (no parameters)
    - Set correct content type

Explicit Registration - OP View

1. Check if Entity Configuration or Trust Chain
2. Collect + Verify / Verify Trust Chain; choose one
3. Verify signature on request
4. If previous registration exists, invalidate it
5. Construct Entity Statement so that
   - The metadata_policies when applied to the RP's metadata statement results in acceptable metadata
   - Set trust_anchor_id to the selected TA's id.
6. Sign and return response
   - Set correct content type

Explicit Registration - RP again

1. Verify correctness of the received Entity Statement
   - trust_anchor_id must be reachable from one of the selected authority_hints
2. Apply metadata_policies from the Trust Chain from the RP to the TA to its own metadata
3. Apply metadata_policies from the OP
4. Store metadata to use
5. Evaluate OP's metadata using the relevant Trust Chain
6. Store OP's metadata to use

Explicit Registration - Expiration

- OIDC fed explicit client registration is not valid forever
  - Entity Statements all have expiration times
- RP must expect that the registration becomes invalidated at any time
  - Re-register
- RP MUST/SHOULD regularly check the Trust Chain between OP and TA
- OP MUST regularly check the Trust Chain between RP and TA and that the signature on the registration request did not expire

Icon set 1 (use them as much you can)


Users: Arts and Education


Users: Earth Observation


Users: Energy


Users: Health and Medicine


Users: Space


Users: Physics


User Groups


Efficiency


Cloud Services


Strategy and Vision


Schedule


TFs and SIGs


Storage


Real-Time Communications


Professional Services


Connectivity and Network

GÉANT Logo


GÉAN TPeop le


GÉANT Services


GÉANT Network


GÉANT Community


EU Logo


Research Programme s


Education Programme s

Note: There are also a number of icons to use as well…please let us know any other design requests

Icon set 2 (use them as much you can)


Latin America region


CAREN region


Asia Pacific region


AC2 region


EUMED region


EaPConnect region


North America


Europe


Locatio n


Glue


Vision


Strategy & Vision


Campus


Service cogs 1


Service cogs 2


Finance


Pilot


Production


Developer


Journal/Library


Legal


NREN


Events


Compass

Note: There are also a number of icons to use as well…please let us know any other design requests

Exascale Network

Trust & Identity

Funding

Virus

Chemistry

Email

Contaminated folder

Parameters

User

Video

Like

Alarm

Info

Access

Colour Palette

Primary

PMS | 303 CMYK | 100 0 0 76 RGB | 0 68 97 Hex | #004461	PMS | 1925 CMYK | 0 100 55 0 RGB | 237 21 87 Hex | #ED1556
Teal is the lead primary colour, its neutrality providing a strong background for accent and secondary colours.	Crimson is the accent primary colour, its brightness providing a strong contrast to the teal and a lead for the secondary colours.

Secondary

PMS | 259 CMYK | 60 80 0 10 RGB | 97 50 131 Hex | #6132	PMS | 1665 CMYK | 0 69 100 0 RGB | 238 80 25 Hex | #EE5019	PMS | 233 CMYK | 10 100 0 0 RGB | 210 0 125 Hex | #D2007D	PMS | 7543 CMYK | 7 0 3 30 RGB | 166 179 179 Hex | #A6B3B3

The secondary colours have been selected to complement the primary colours. These secondary colours should be used sparingly and only used in addition to, and where they will not interfere with the primary colours.

Thank you

Any questions?

The research leading to these results has received funding from
the European Union's Horizon 2020 research and innovation programme under Grant Agreement No. 856726 (GN4-3).