Introduction to Verifiable Public Registry

Fabrice Rochette
https://www.linkedin.com/in/fabricerochette/

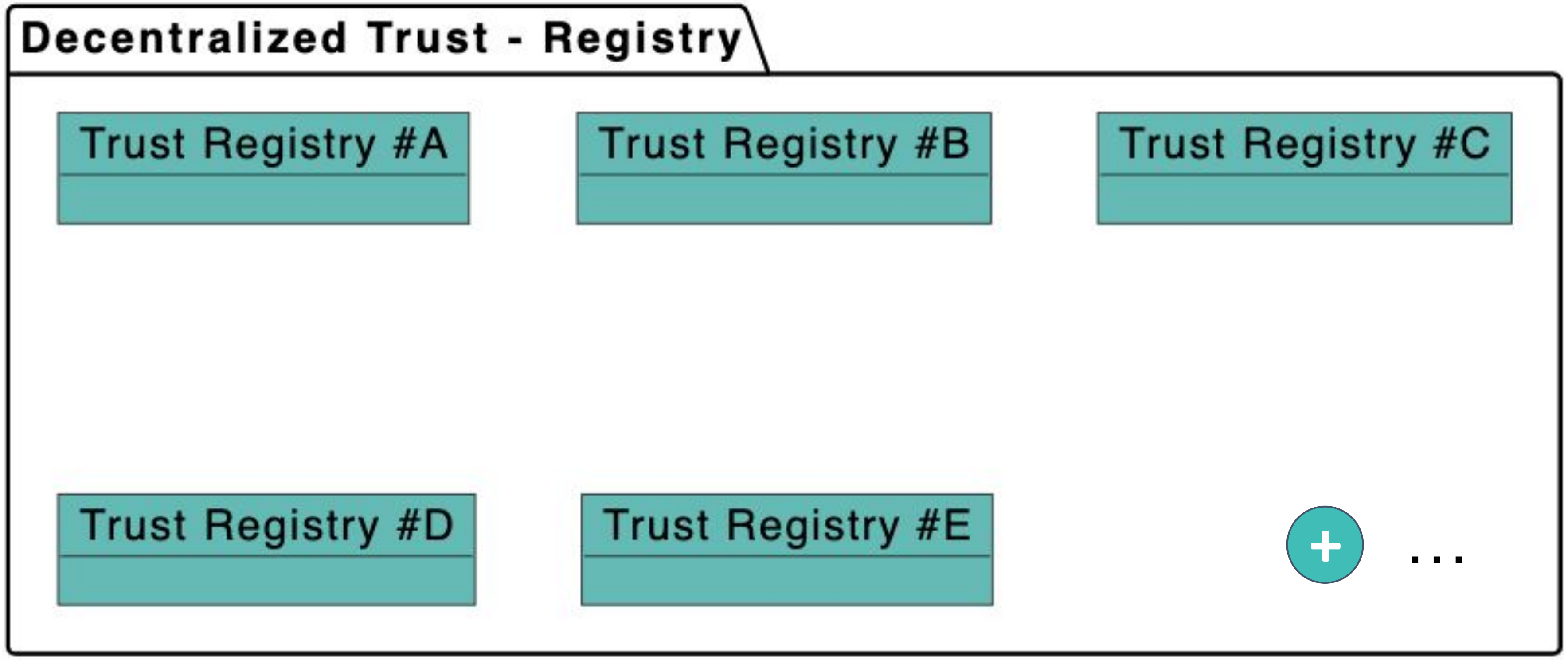
IIWXXXIX Fall 2024

PRIVACY NOTICE

What is a Verifiable Public Registry?

A VPR is a public Registry of Trust Registries

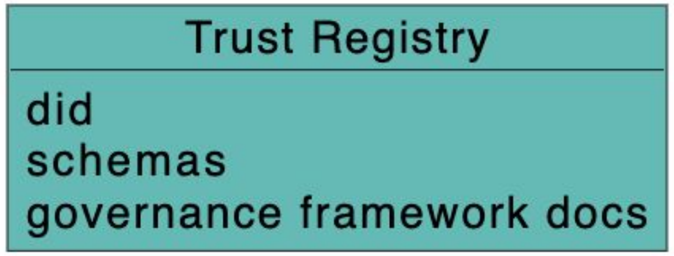
Anyone can create a Trust Registry in a VPR.



Trust Registries

Each **Trust Registry** is identified by a **resolvable DID**, and provides, at least:

- **Governance Framework** document(s).
- Zero or more **Credential Schemas**.



A VPR doesn’t care about the DID methods used because DT resolution is performed outside the VPR.

In a VPR, you can use **any DID method**.

Credential Schemas

Credential Schemas

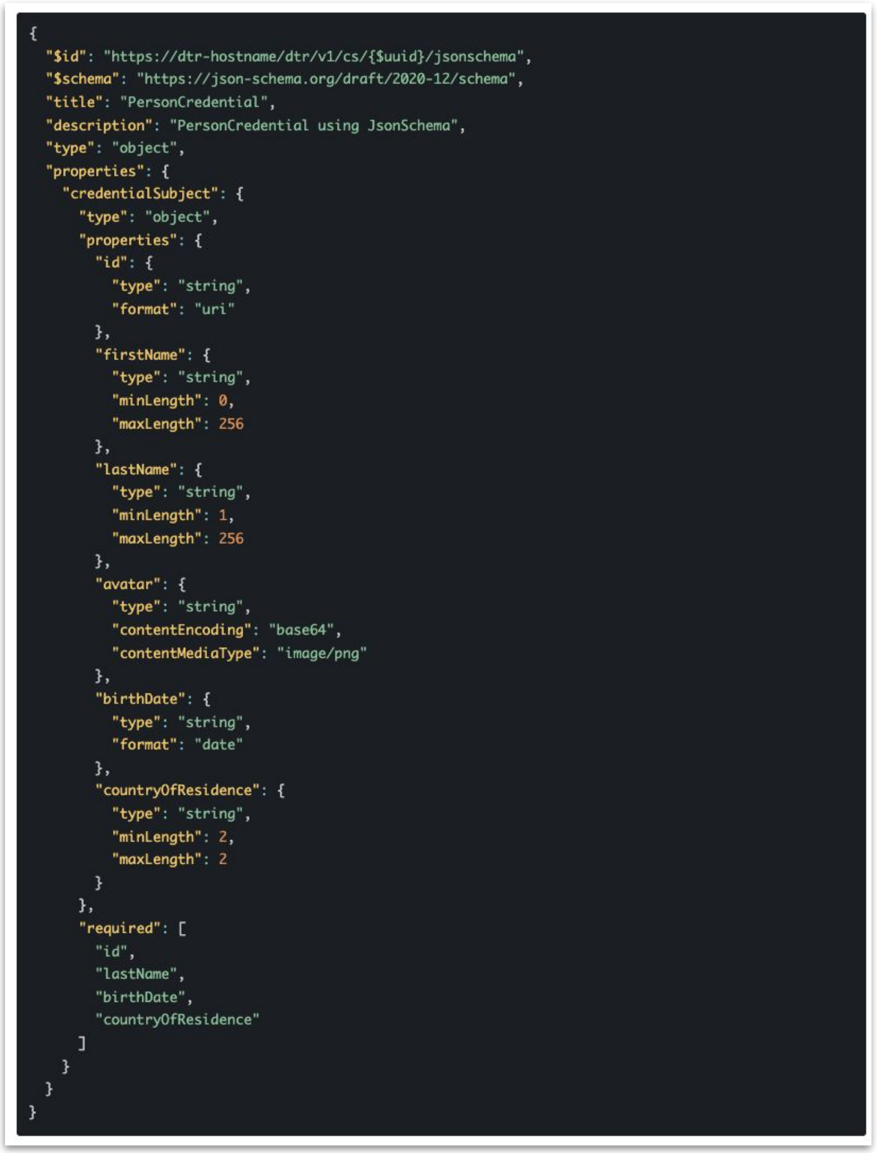
They are created and controlled by **Trust Registries**.

Credential Schemas include:

- Configuration information;
- A **Json Schema** of the Credential Schema.

A **Credential Schema** is immutable.

Credential Schema Permissions (CSPs) define who can perform actions related to the **Credential Schema**, such as onboarding **issuers** and **verifiers**, **issue** or **verify** credentials. CSPs define optional business rules.

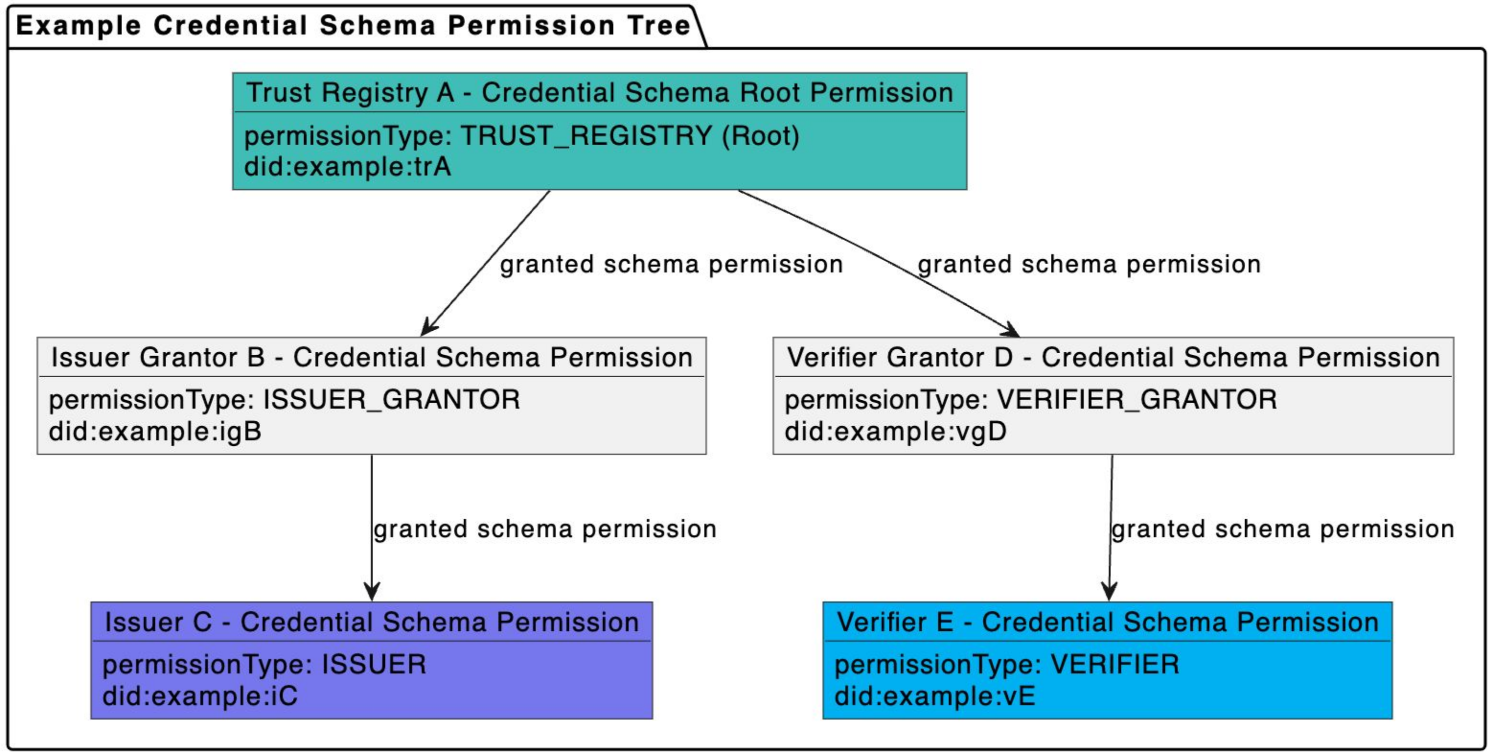


```json
{
  "$id": "https://dtr-hostname/dtr/v1/cs/{$uuid}/jsonschema",
  "$schema": "https://json-schema.org/draft/2020-12/schema",
  "title": "PersonCredential",
  "description": "PersonCredential using JsonSchema",
  "type": "object",
  "properties": {
    "credentialSubject": {
      "type": "object",
      "properties": {
        "id": {
          "type": "string",
          "format": "uri"
        },
        "firstName": {
          "type": "string",
          "minLength": 0,
          "maxLength": 256
        },
        "lastName": {
          "type": "string",
          "minLength": 1,
          "maxLength": 256
        },
        "avatar": {
          "type": "string",
          "contentEncoding": "base64",
          "contentMediaType": "image/png"
        },
        "birthDate": {
          "type": "string",
          "format": "date"
        },
        "countryOfResidence": {
          "type": "string",
          "minLength": 2,
          "maxLength": 2
        }
      },
      "required": [
        "id",
        "lastName",
        "birthDate",
        "countryOfResidence"
      ]
    }
  }
}
```

Credential Schema Permissions

Each Credential Schema has its own Permission tree

Credential Schema defines which Permission Types are allowed

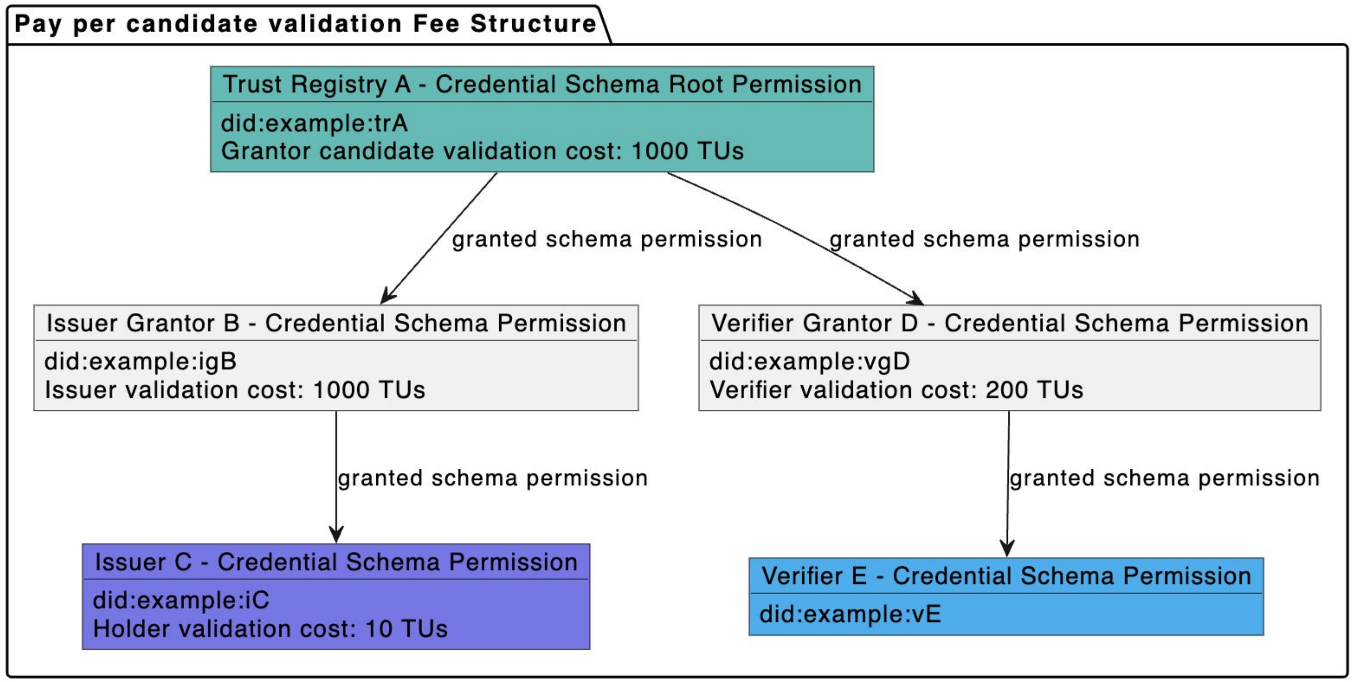

Example Credential Schema Permission Tree
Trust Registry A - Credential Schema Root Permission
permissionType: TRUST_REGISTRY (Root)
did:example:trA
granted schema permission
granted schema permission
Issuer Grantor B - Credential Schema Permission
permissionType: ISSUER_GRANTOR
did:example:igB
Verifier Grantor D - Credential Schema Permission
permissionType: VERIFIER_GRANTOR
did:example:vgD
granted schema permission
granted schema permission
Issuer C - Credential Schema Permission
permissionType: ISSUER
did:example:iC
Verifier E - Credential Schema Permission
permissionType: VERIFIER
did:example:vE

Permission Type	Description
Trust Registry	Create and control Credential Schemas. Grant other roles.
Issuer Grantor	Grant Issuer permissions to candidate issuers
Verifier Grantor	Grant Verifier permissions to candidate verifiers
Issuer	Can issue credentials of this schema
Verifier	Can request presentation of credentials of this schema

Validation Process: to create new CSPs and/or issue VCs

To get granted a CSP, an Applicant must run a Validation process

Applicant starts the Validation Process by selecting a Validator CSP. Validator CSP defines required fees.



Pay per candidate validation Fee Structure
Trust Registry A - Credential Schema Root Permission
did:example:trA
Grantor candidate validation cost: 1000 TUs
granted schema permission
granted schema permission
Issuer Grantor B - Credential Schema Permission
did:example:igB
Issuer validation cost: 1000 TUs
Verifier Grantor D - Credential Schema Permission
did:example:vgD
Verifier validation cost: 200 TUs
granted schema permission
granted schema permission
Issuer C - Credential Schema Permission
did:example:iC
Holder validation cost: 10 TUs
Verifier E - Credential Schema Permission
did:example:vE

+

Code	Description	Rate
TD	**Trust Deposit**	20%

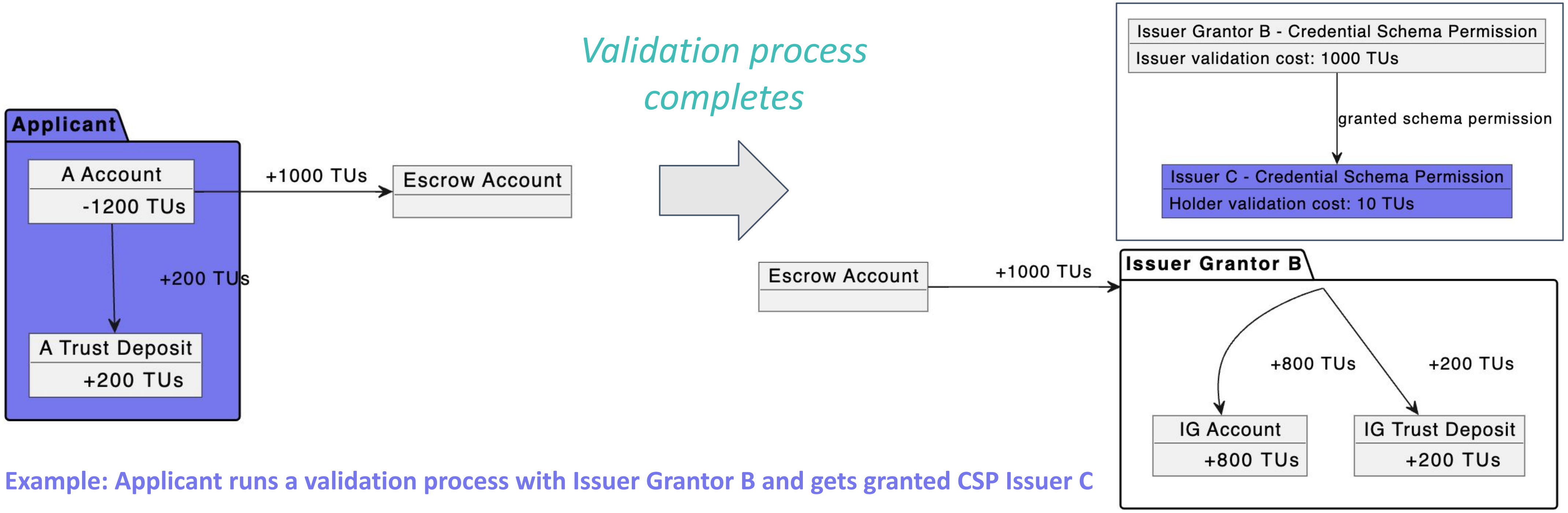
In this example:

- An Applicant will need to pay 1,000 * (1 + TD) = 1,200 TUs to run a validation process with **Issuer Grantor B** and get granted an **ISSUER** CSP **Issuer C** for this **Credential Schema** of **Trust Registry A**

Validation Process: to create new CSPs and/or issue VCs

Validation Process runs, and optional fees are distributed.

- **Validation** is started, fees paid by **Applicant** are escrowed
- **Applicant** connects to the **DT-Service (DTS)** provided by **Validator** (the **DID** registered in the **Validator's** CSP). They exchange information for completing the **Validation** process**.**
- When **Validation** process completes, Applicant CSP is created (and/or a credential is issued), then fees are distributed.

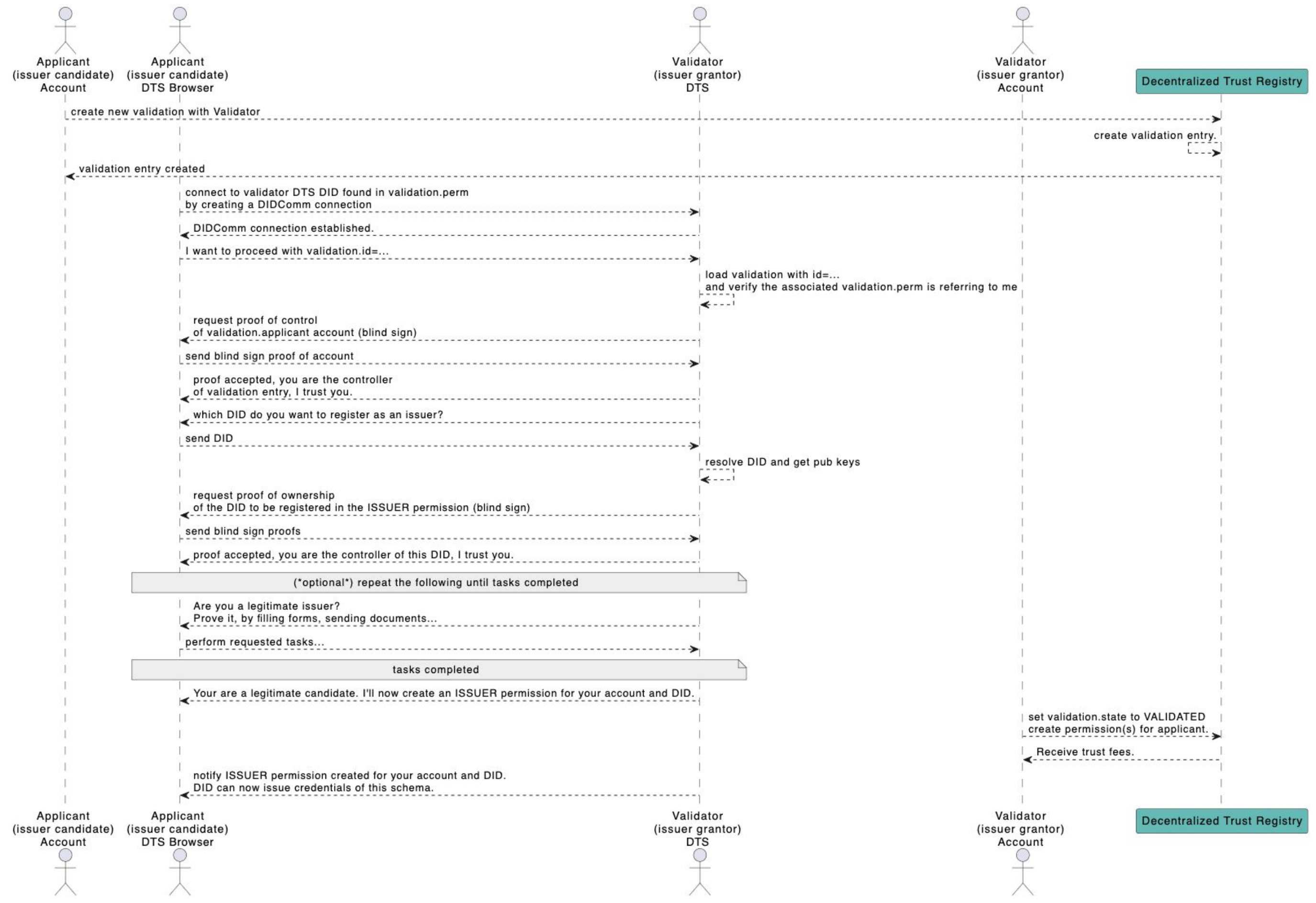


Validation process completes
Applicant
A Account
-1200 TUs
+1000 TUs
Escrow Account
+200 TUs
A Trust Deposit
+200 TUs
Issuer Grantor B - Credential Schema Permission
Issuer validation cost: 1000 TUs
granted schema permission
Issuer C - Credential Schema Permission
Holder validation cost: 10 TUs
Escrow Account
+1000 TUs
Issuer Grantor B
+800 TUs
+200 TUs
IG Account
+800 TUs
IG Trust Deposit
+200 TUs

Example: Applicant runs a validation process with Issuer Grantor B and gets granted CSP Issuer C

Validation Process: to create new CSPs and/or issue VCs

Validation Process example



Applicant (issuer candidate) Account
Applicant (issuer candidate) DTS Browser
Validator (issuer grantor) DTS
Validator (issuer grantor) Account
Decentralized Trust Registry

create new validation with Validator

create validation entry.

validation entry created

connect to validator DTS DID found in validation.perm
by creating a DIDComm connection

DIDComm connection established.

I want to proceed with validation.id=...

load validation with id=...
and verify the associated validation.perm is referring to me

request proof of control
of validation.applicant account (blind sign)

send blind sign proof of account

proof accepted, you are the controller
of validation entry, I trust you.

which DID do you want to register as an issuer?

send DID

resolve DID and get pub keys

request proof of ownership
of the DID to be registered in the ISSUER permission (blind sign)

send blind sign proofs

proof accepted, you are the controller of this DID, I trust you.

(*optional*) repeat the following until tasks completed

Are you a legitimate issuer?
Prove it, by filling forms, sending documents...

perform requested tasks...

tasks completed

Your are a legitimate candidate. I'll now create an ISSUER permission for your account and DID.

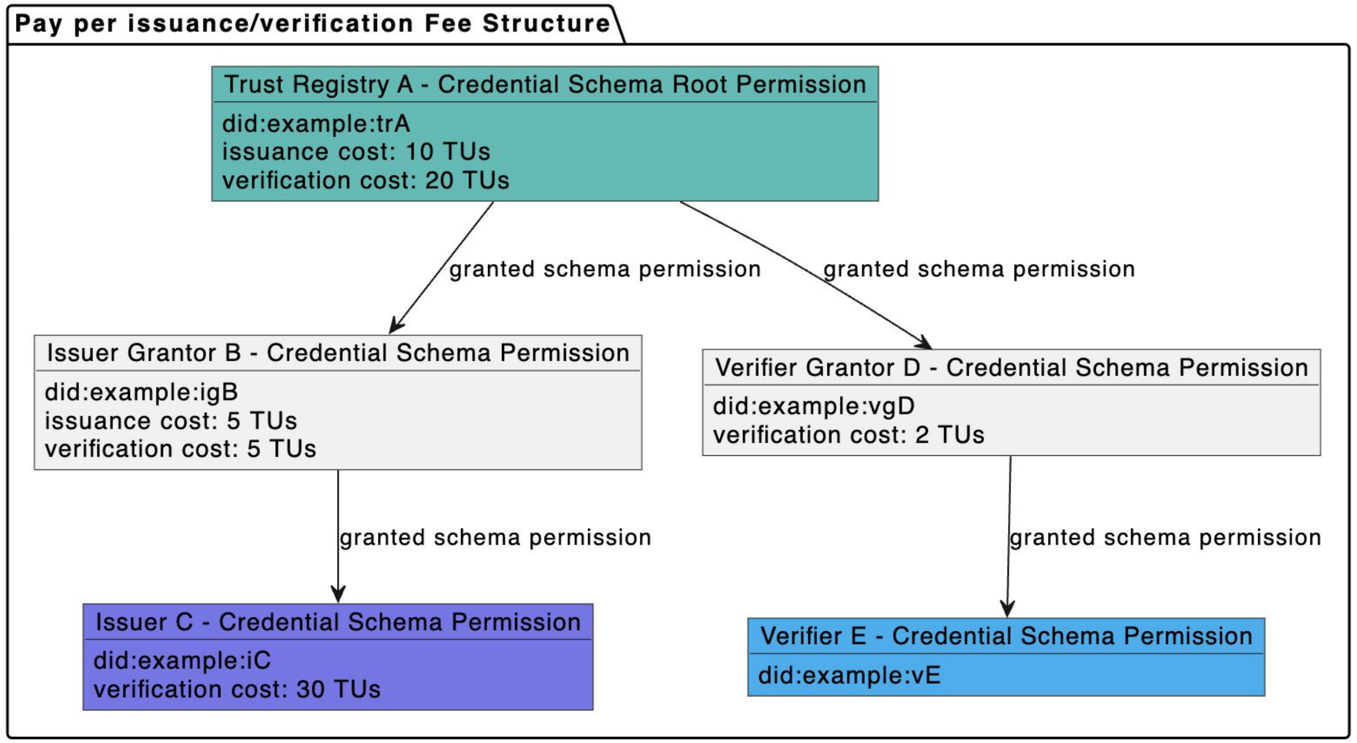
set validation.state to VALIDATED
create permission(s) for applicant.

Receive trust fees.

notify ISSUER permission created for your account and DID.
DID can now issue credentials of this schema.

Pay per issuance/verification: applying CSP rules

CSPs provide a flexible pay per issuance/verification model


Pay per issuance/verification Fee Structure
Trust Registry A - Credential Schema Root Permission
did:example:trA
issuance cost: 10 TUs
verification cost: 20 TUs
granted schema permission
granted schema permission
Issuer Grantor B - Credential Schema Permission
did:example:igB
issuance cost: 5 TUs
verification cost: 5 TUs
Verifier Grantor D - Credential Schema Permission
did:example:vgD
verification cost: 2 TUs
granted schema permission
granted schema permission
Issuer C - Credential Schema Permission
did:example:iC
verification cost: 30 TUs
Verifier E - Credential Schema Permission
did:example:vE

+

Code	Description	Rate
UAR	**User Agent Rate**, for rewarding Apps and Browser and Services that enforce the trust layer	10%
WUAR	**Wallet User Agent Rate**, for rewarding Wallets and Services that enforce the trust layer	10%
TD	**Trust Deposit**	20%

In this example:

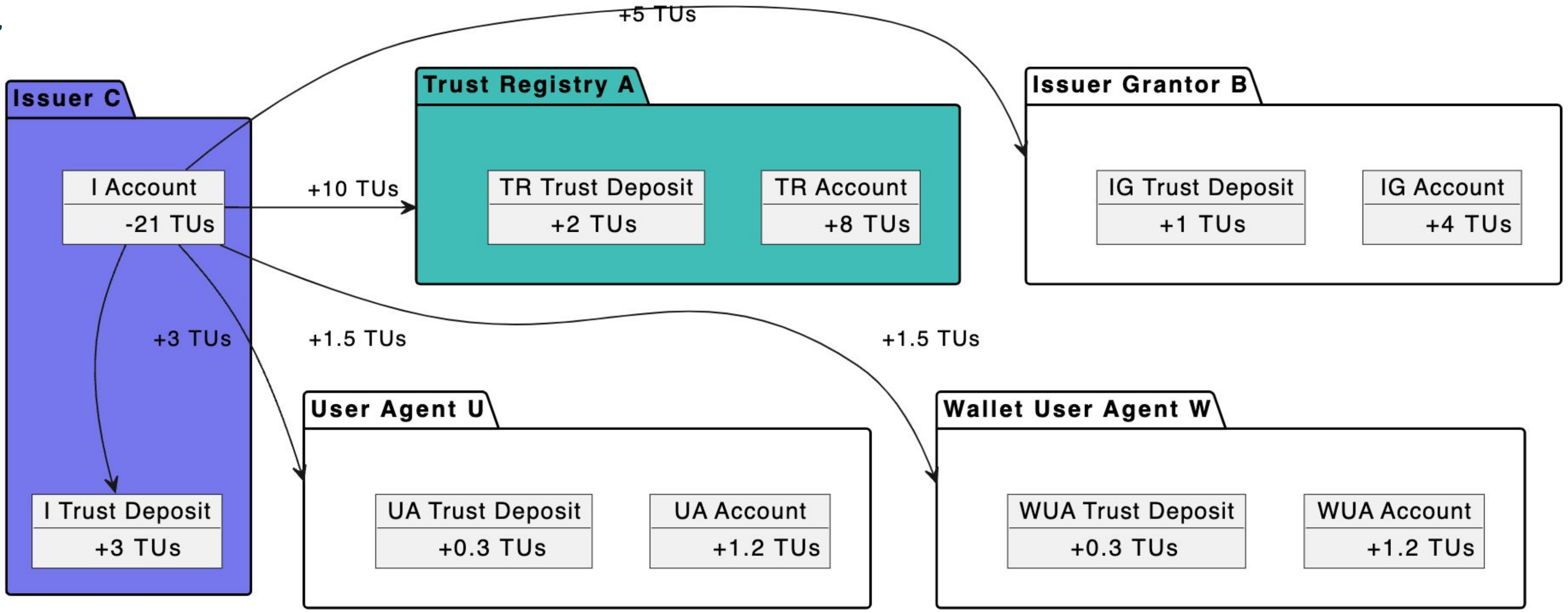
- Total paid by **Issuer C** for issuing a credential: (10 + 5) * (1 + UAR + WUAR + TD) = **21 TUs**
- Total paid by **Verifier E** for verifying a credential: (20 + 5 + 2 + 30) * (1 + UAR + WUAR + TD) = **79.8 TUs**

Pay per issuance/verification: applying CSP rules

A flexible pay per issuance/verification model that rewards all participants

- If fees>0, Issuer must create a transaction else **DT compliant wallet** will not accept the credential
- Fee distribution is automatically handled by VPR
- **Privacy Preserving** for Holder

Fees distribution:



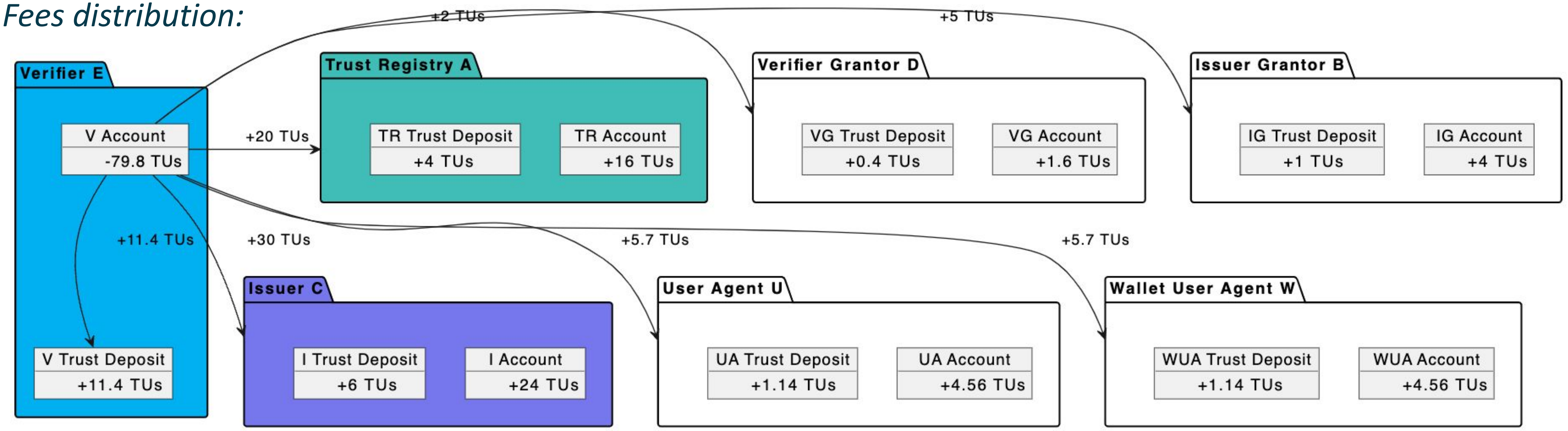
Credential Issuance

Pay per issuance/verification: applying CSP rules

A flexible pay per issuance/verification model that rewards all participants

- If fees>0, Verifier must create a transaction else **DT compliant wallet** will not accept the presentation request
- Fee distribution is automatically handled by DTR network
- **Privacy Preserving** for Holder

Fees distribution:

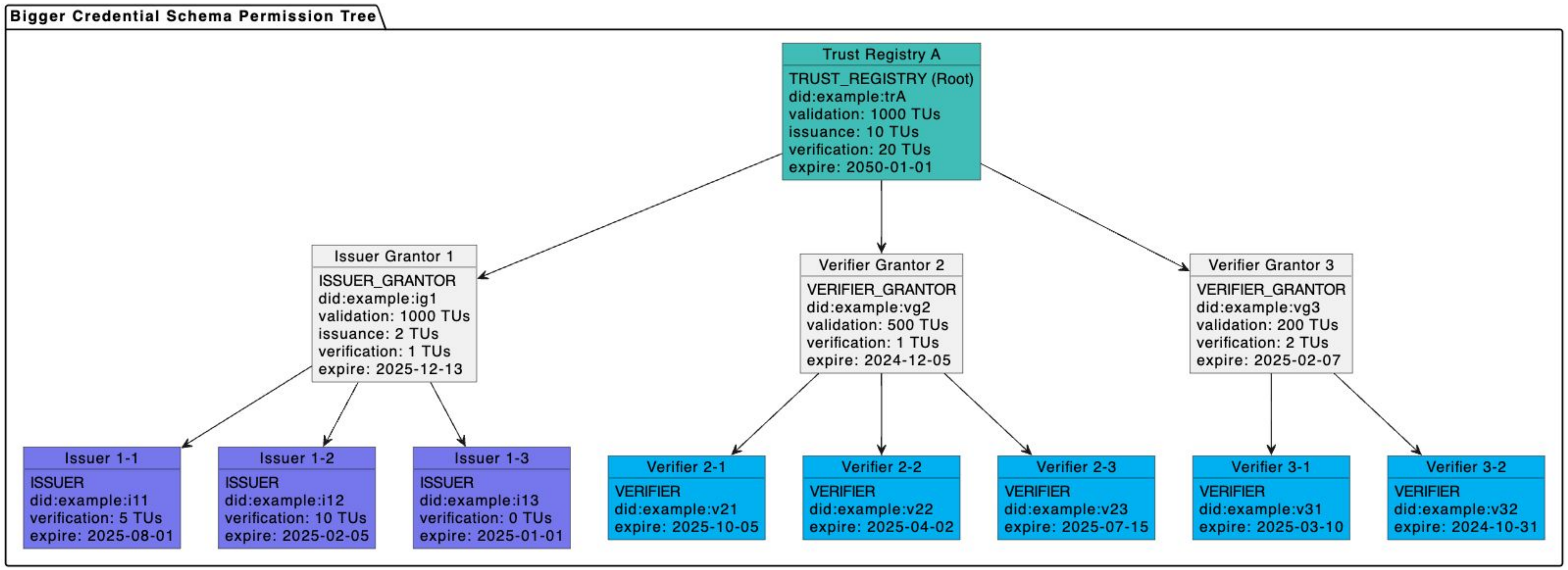


Credential Verification

Credential Schema Permissions: Summary

A decentralized way of controlling permissions

- All participants must comply with the Trust Registry Governance Framework
- **Validation** processes are needed for creating and maintaining **CSPs**
- For issuing (resp. verifying) credentials, **Issuer** (resp. **Verifier**) may have to **pay fees**.

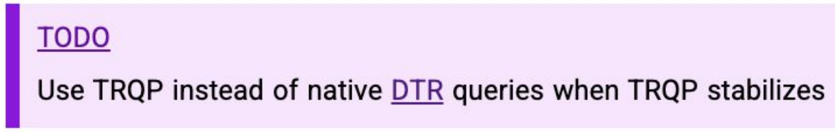


Credential Schema Permissions: Query examples

Issuer

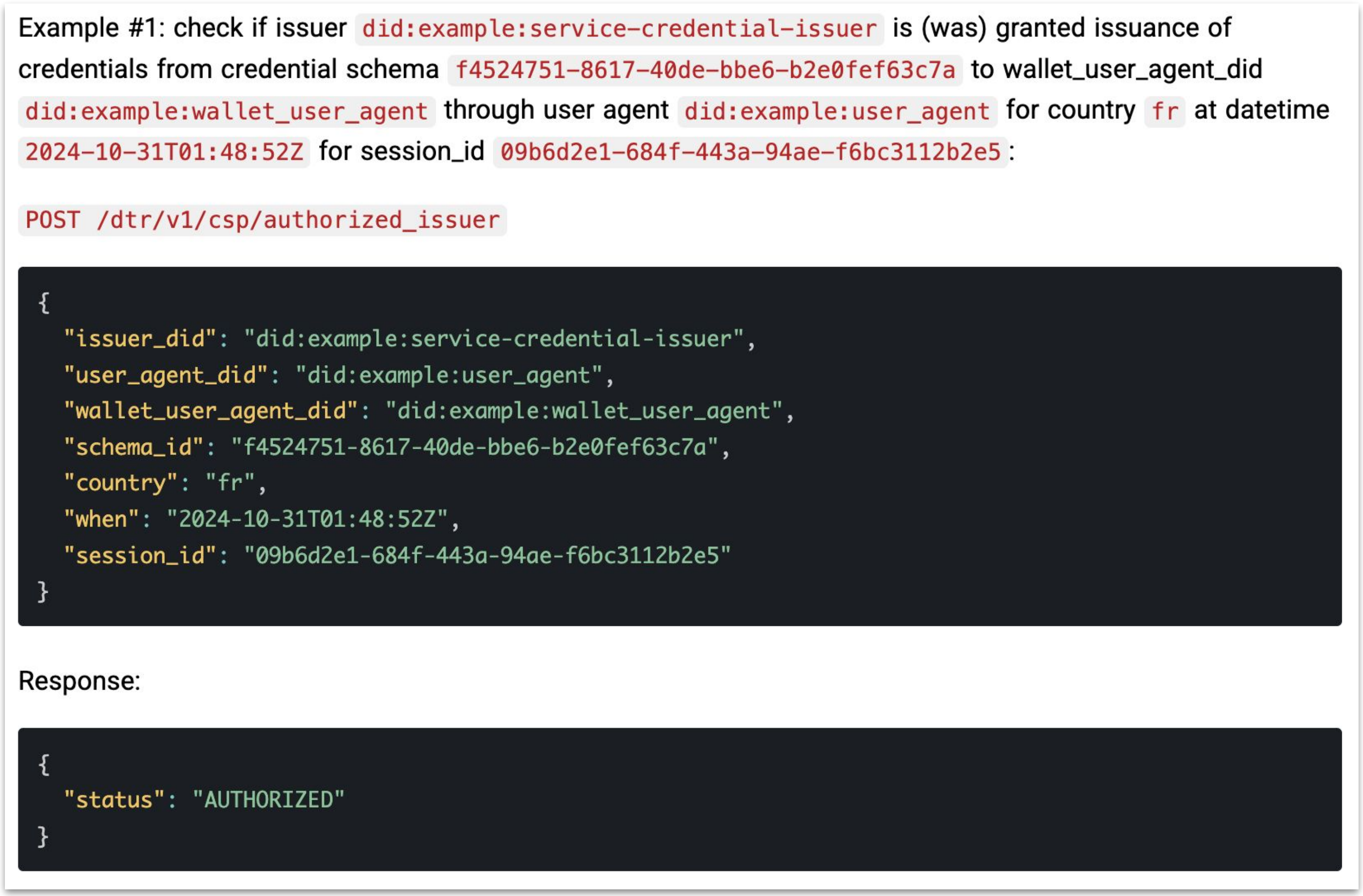
A query must be performed by a **DT compliant User Agent or Service** (**DT-UA, DT-S**) before accepting a credential from a given **Issuer**.

Spec: https://verana-labs.github.io/verifiable-trust-spec/



TODO

Use TRQP instead of native DTR queries when TRQP stabilizes



Example #1: check if issuer `did:example:service-credential-issuer` is (was) granted issuance of credentials from credential schema `f4524751-8617-40de-bbe6-b2e0fef63c7a` to wallet_user_agent_did `did:example:wallet_user_agent` through user agent `did:example:user_agent` for country `fr` at datetime `2024-10-31T01:48:52Z` for session_id `09b6d2e1-684f-443a-94ae-f6bc3112b2e5`:

`POST /dtr/v1/csp/authorized_issuer`

```
{
  "issuer_did": "did:example:service-credential-issuer",
  "user_agent_did": "did:example:user_agent",
  "wallet_user_agent_did": "did:example:wallet_user_agent",
  "schema_id": "f4524751-8617-40de-bbe6-b2e0fef63c7a",
  "country": "fr",
  "when": "2024-10-31T01:48:52Z",
  "session_id": "09b6d2e1-684f-443a-94ae-f6bc3112b2e5"
}
```

Response:

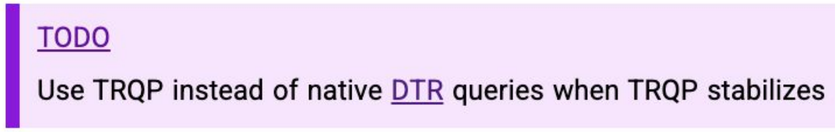
```
{
  "status": "AUTHORIZED"
}
```

Credential Schema Permissions: Query examples

Verifier

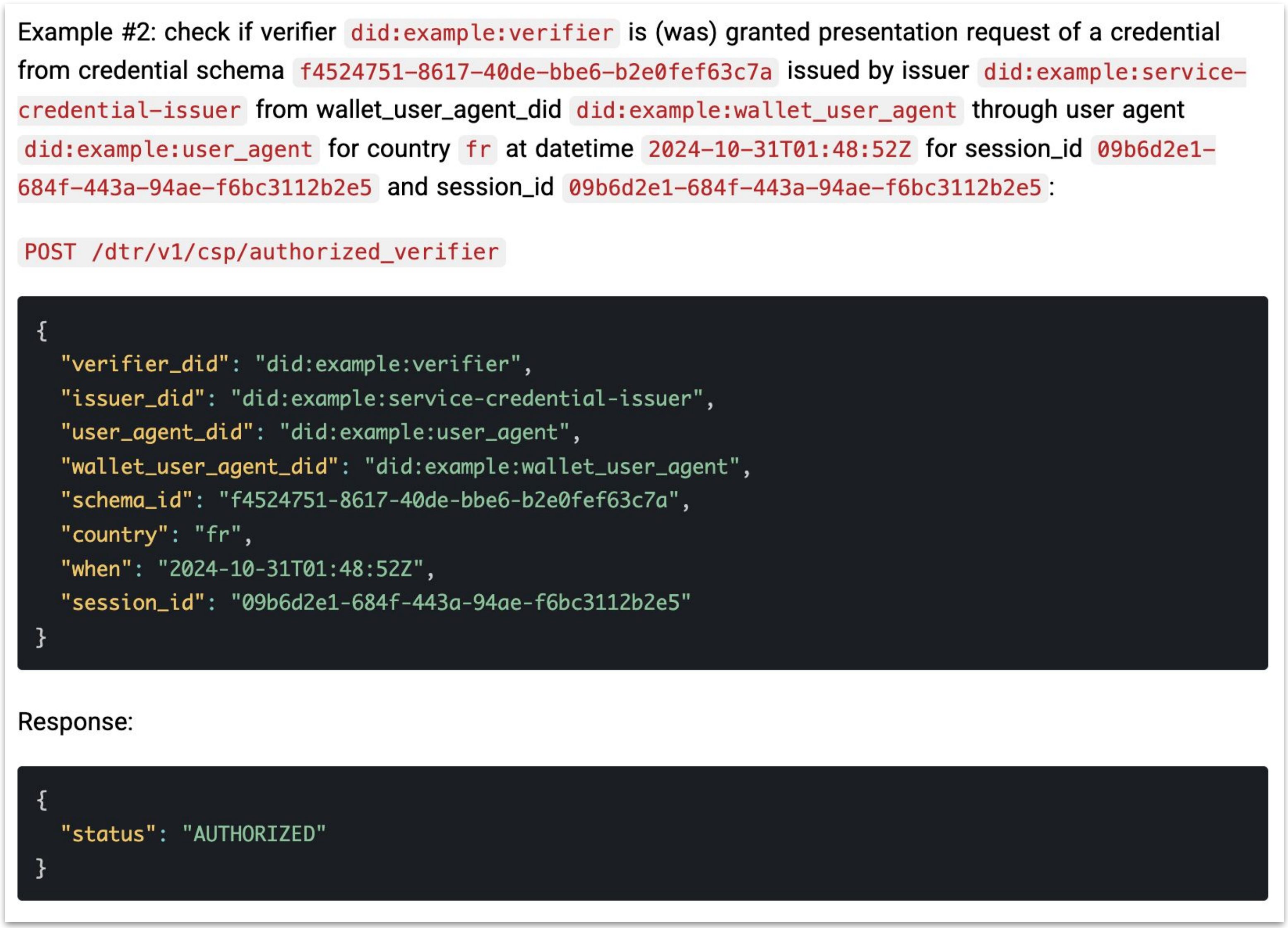
A query must be performed by a **DT compliant User Agent or Service** (**DT-UA, DT-S**) before accepting presenting a credential to a given **Verifier**.

Spec: https://verana-labs.github.io/verifiable-trust-spec/



TODO

Use TRQP instead of native DTR queries when TRQP stabilizes



Example #2: check if verifier `did:example:verifier` is (was) granted presentation request of a credential from credential schema `f4524751-8617-40de-bbe6-b2e0fef63c7a` issued by issuer `did:example:service-credential-issuer` from wallet_user_agent_did `did:example:wallet_user_agent` through user agent `did:example:user_agent` for country `fr` at datetime `2024-10-31T01:48:52Z` for session_id `09b6d2e1-684f-443a-94ae-f6bc3112b2e5` and session_id `09b6d2e1-684f-443a-94ae-f6bc3112b2e5`:

`POST /dtr/v1/csp/authorized_verifier`

```
{
  "verifier_did": "did:example:verifier",
  "issuer_did": "did:example:service-credential-issuer",
  "user_agent_did": "did:example:user_agent",
  "wallet_user_agent_did": "did:example:wallet_user_agent",
  "schema_id": "f4524751-8617-40de-bbe6-b2e0fef63c7a",
  "country": "fr",
  "when": "2024-10-31T01:48:52Z",
  "session_id": "09b6d2e1-684f-443a-94ae-f6bc3112b2e5"
}
```

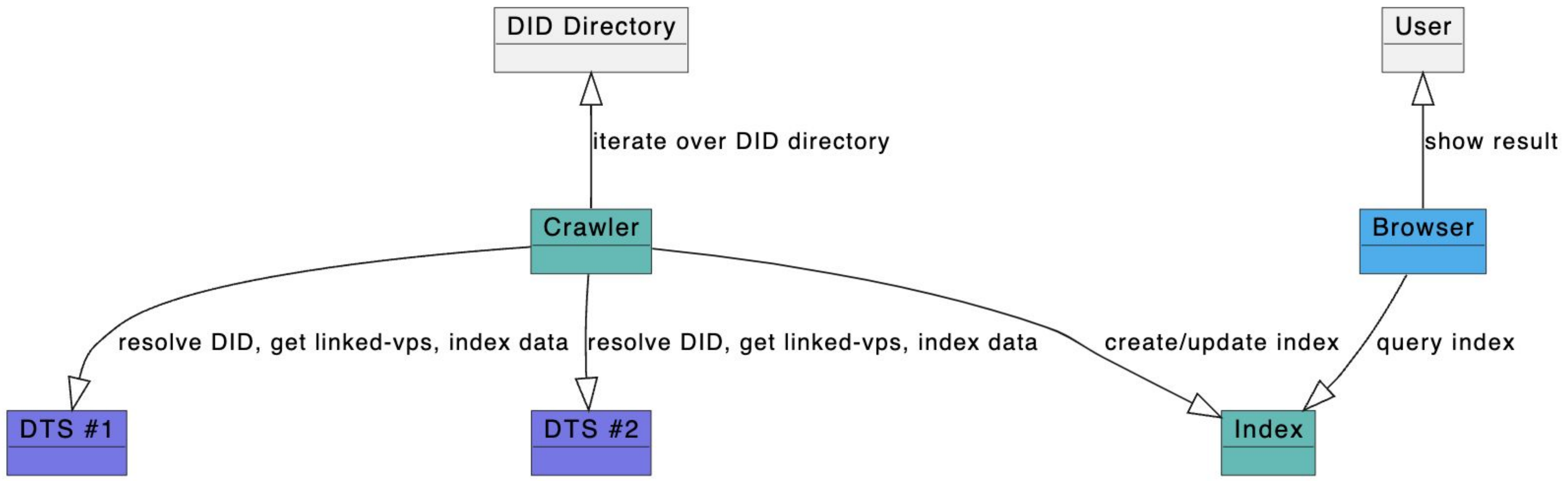
Response:

```
{
  "status": "AUTHORIZED"
}
```

DID Directory

A Directory of verifiable services

The DID directory is a **public database of DIDs** that can be used by **crawlers** to build an index of Decentralized Trust - Services (DTSs). Crawlers simply need to iterate over the DID Directory, and for each DID, try to resolve its DID Document, and dereference all interesting information as explained in the **Verifiable Trust Specification**.

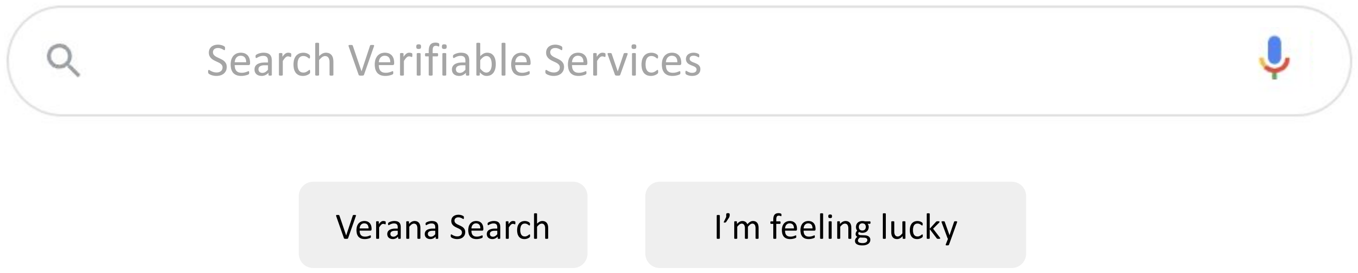

DID Directory
User
iterate over DID directory
show result
Crawler
Browser
resolve DID, get linked-vps, index data
resolve DID, get linked-vps, index data
create/update index
query index
DTS #1
DTS #2
Index

Any participant can register a DID in the DID directory.

DID Directory

Used by search engines, apps…

By using the index built by crawlers, a search engine can be provided to users and they can search for **verifiable services** by querying **verifiable metadata**.



Apps can use an index to let user search only for content/services they support. **Example**: a **Social Network App** could work by indexing **Social Channel Verifiable Services** that present a specific credential only, ie a credential issued by the social network app owner.
That’s another business model: purchase a credential to appear in a service.

Verifiable Public Registry - Spec

Contributions? Discussions?

https://github.com/verana-labs/decentralized-trust-registry-spec

2060

Building The Missing Trust Layer

Location

Ahtri tn 12
10151 Tallinn, Estonia

Cra. 13A #86A–42
Bogotá DC, Colombia

Paseo de Recoletos 27-41
Madrid, 28004, Spain

f@2060.io