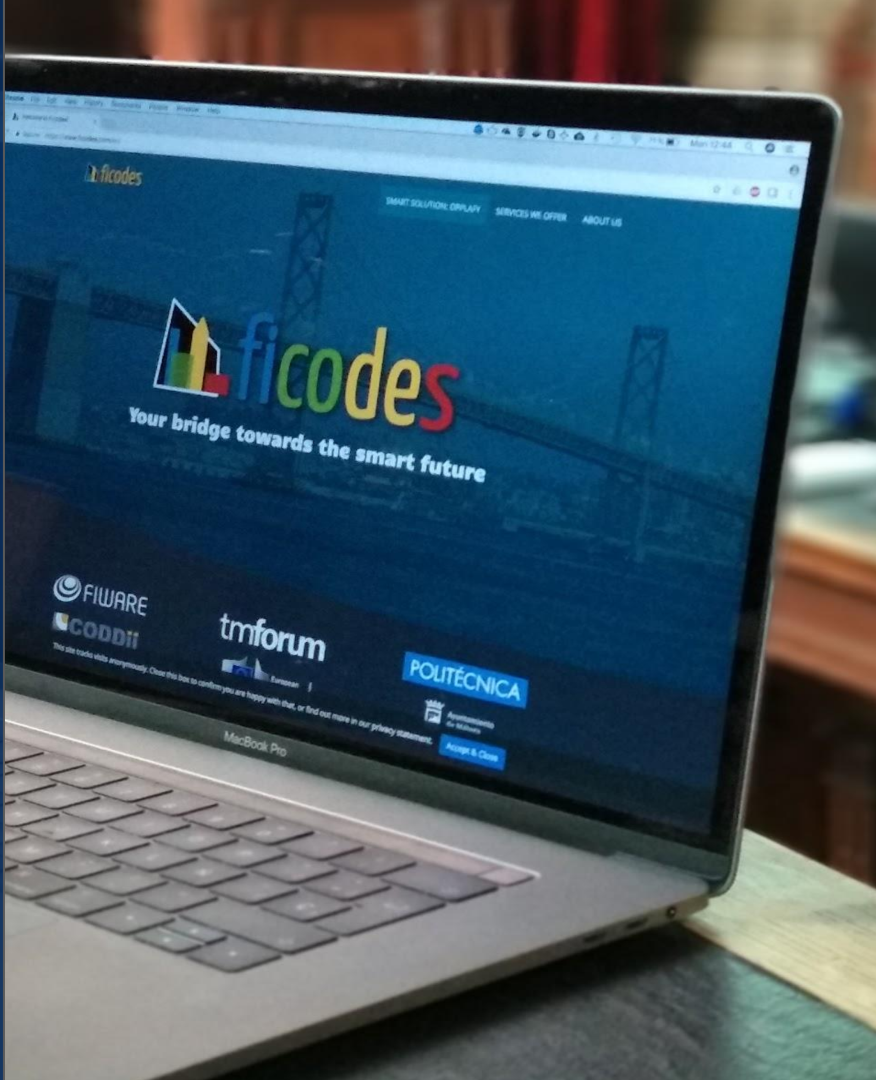
FIWARE Data Spaces

Decentralized Identity and Access Management

Stefan Wiedemann - Senior Software Engineer

Future Internet Consulting and Development Solutions S.L.
C/ Hespérides 5, 28232, Las Rozas de Madrid, Madrid, España

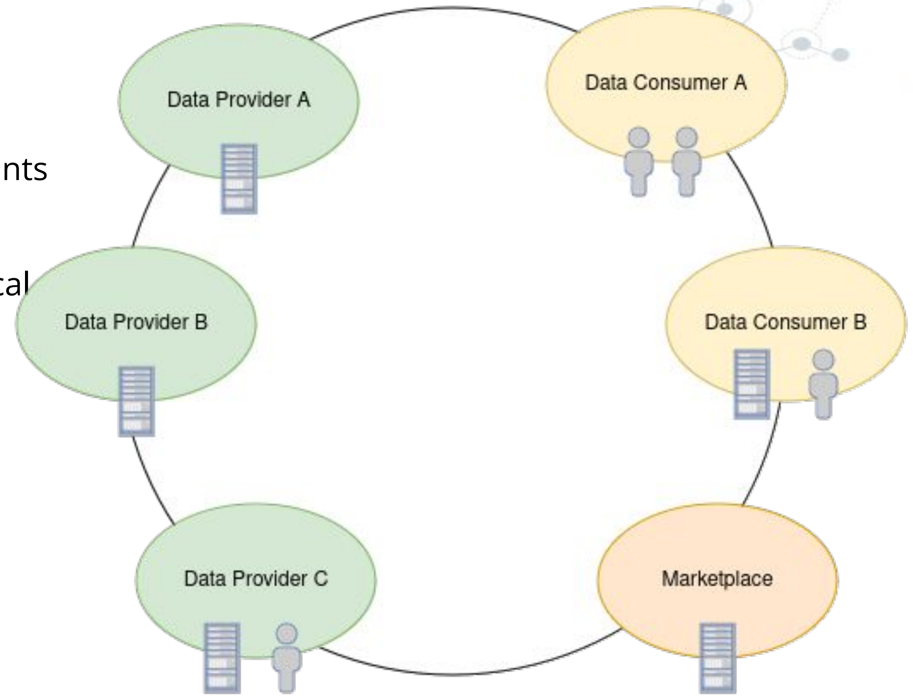


Decentralized IAM

1. Decentralized IAM in Data Spaces - Why?
2. Introduction into Decentralized Identity
    a. Verifiable Credentials
    b. Decentralized Identifiers
3. Authentication and Authorization in a FIWARE Data Space
4. Policy Enforcement with ODRL

Why?

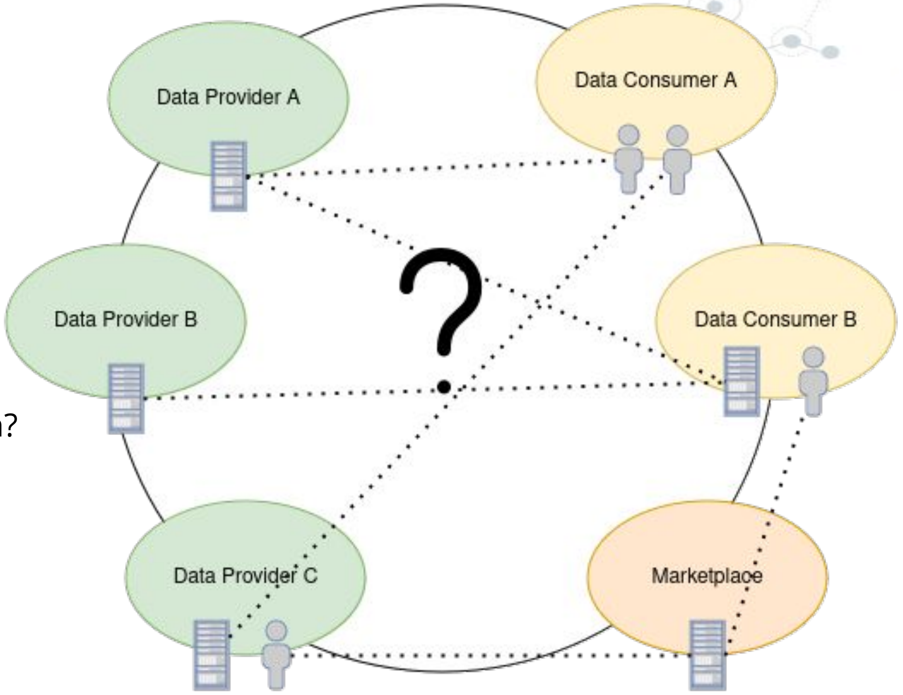
- A Data Spaces consists of different participants in various roles
- The participating organizations have technical and human actors, connecting each other
- No connection beside the Data Spaces between the participants
- No knowledge about each others actors(human or technical)



ficodes

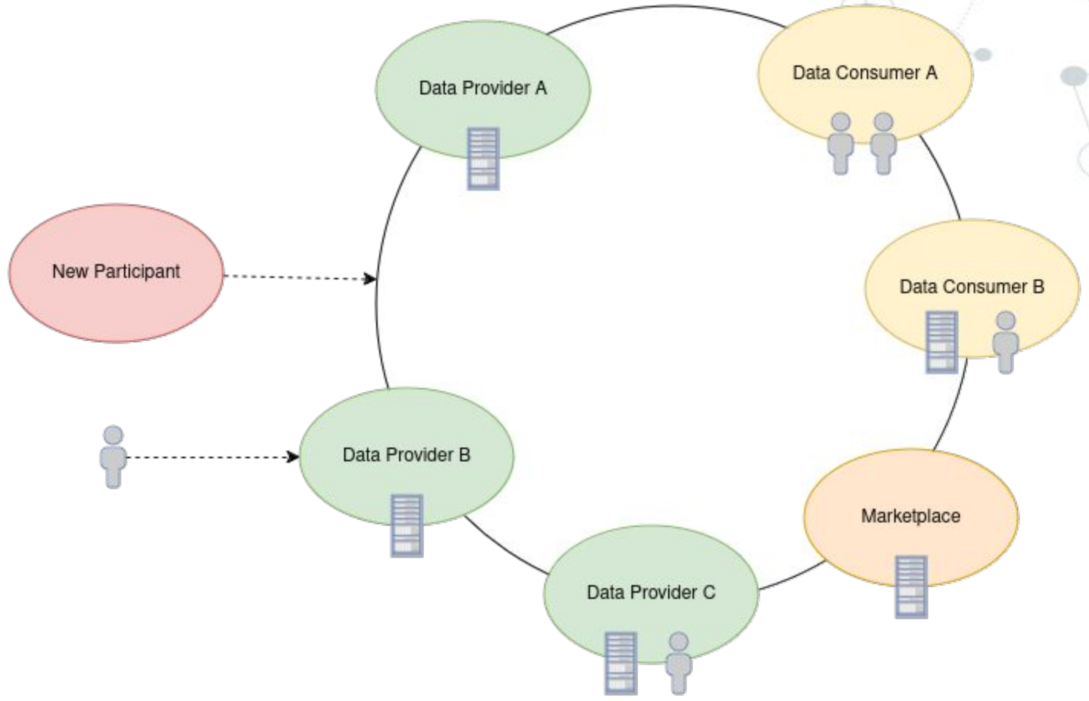
Why?

- How to identify actors?
- How to authorize their actions?
- How to connect them with their organization?



Why?

- New participant joining the Data Space requires a new trust-relationship
- New actor joins one of the participants, needs to be recognized

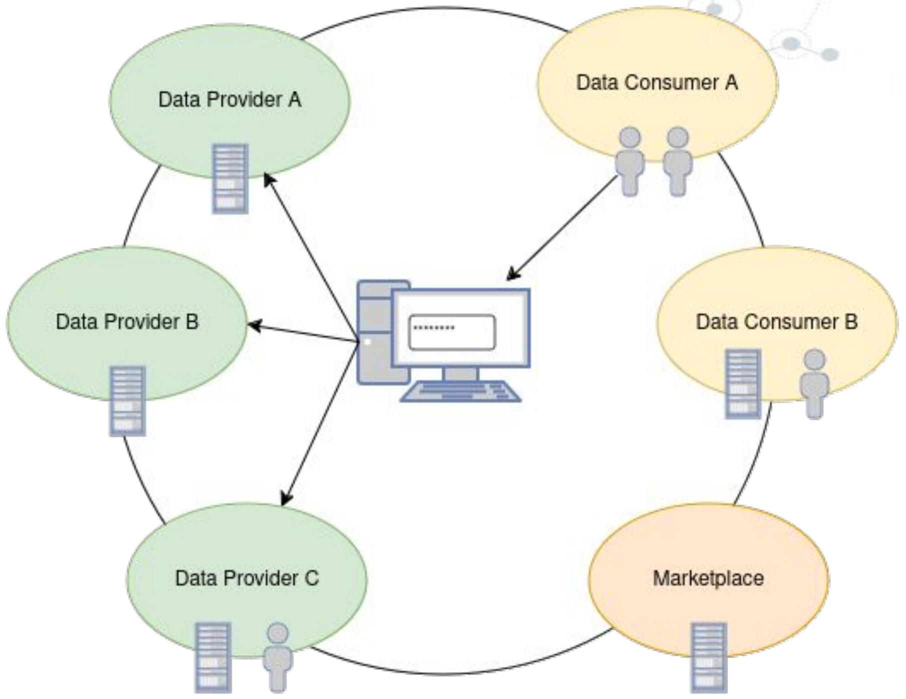


ficodes

Why?

Central Login:

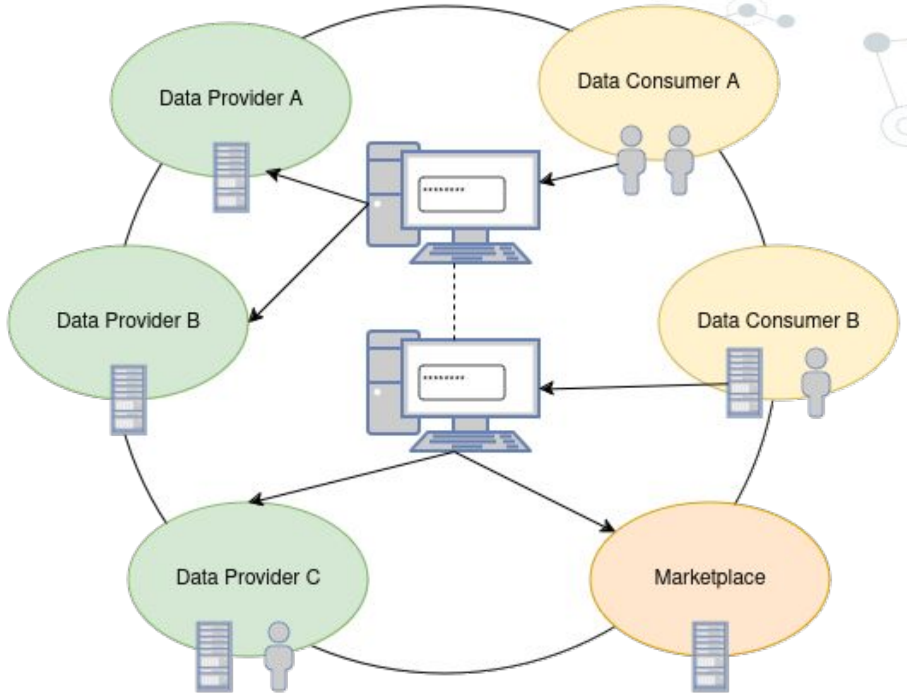
- Easy to use, but all the power at one single entity
- powerful (user) data sink



Why?

Federated Identity:

- Less reliance on a central IDP
- still reliance on “central” providers
- increasing number of providers -> increased complexity

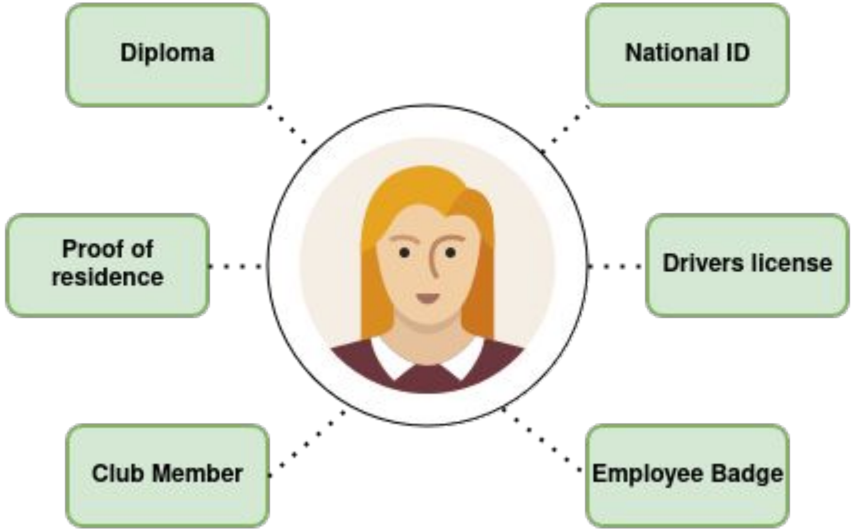


ficodes

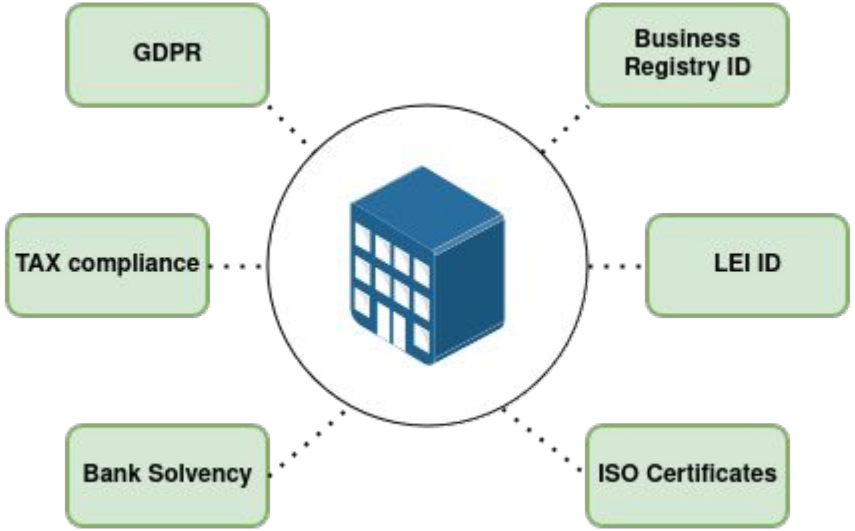
Decentralized Identity

- Copy the good things of the physical world
- Augment it with digital powers
- Preserve the required properties
- Adapt to current legal frameworks, but stay prepared for the future

Identity of a natural person

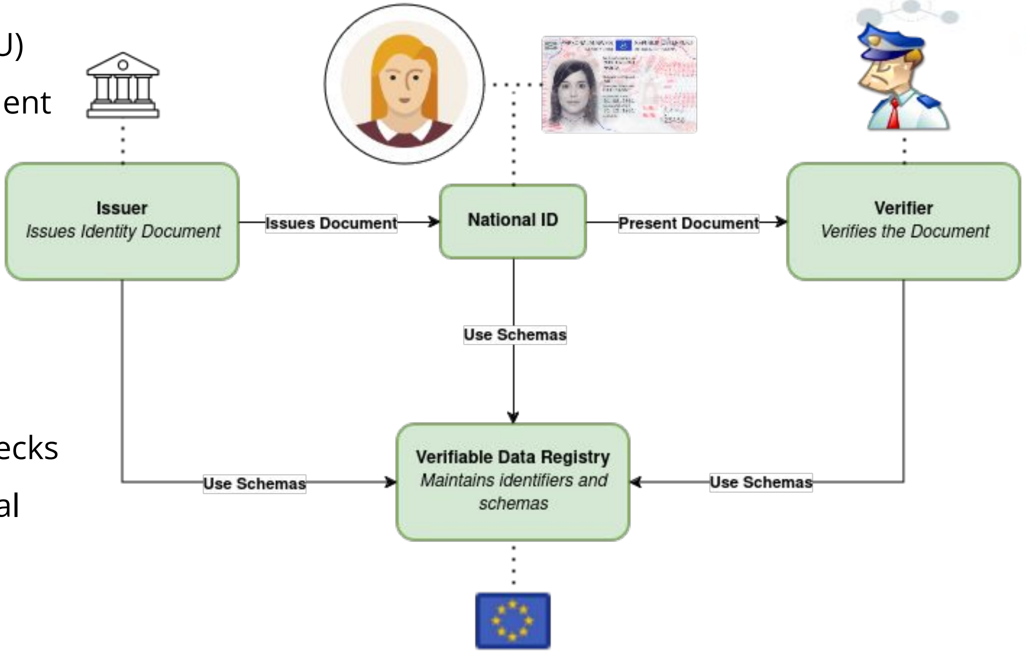


Identity of a legal person



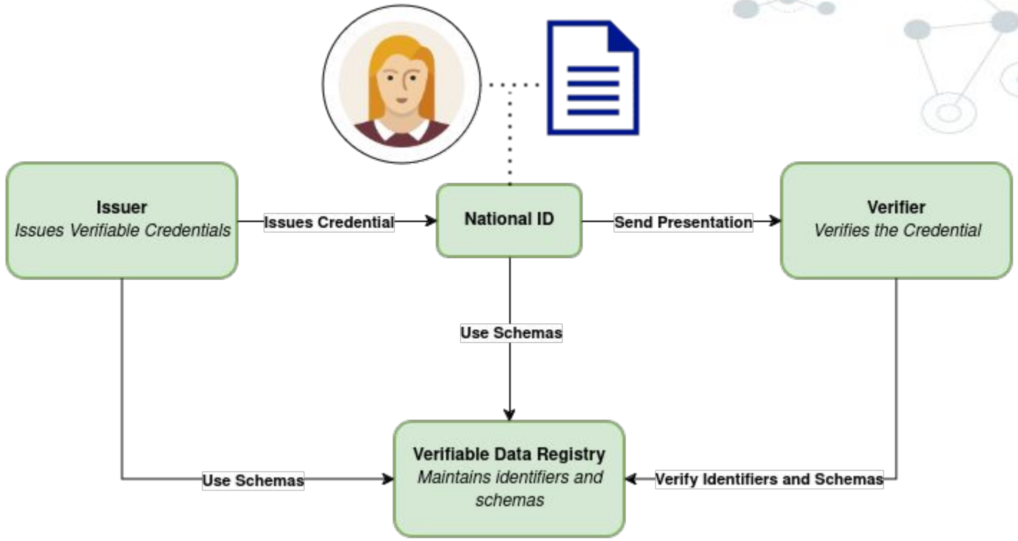
Authentication in the Real World

- Government registered at registry(EU)
- National ID is issued by the government to a citizen
- Follows commonly agreed schemas(defined by the EU)
- Is presented to an authority for authentication
- Verifies authenticity features and checks that its a registered Schema("National ID") and from a registered issuer



W3C Verifiable Credentials

- Issuer is registered at the Verifiable Data Registry
- Issuer issues Credential to the holder
- Holder presents the credential to the verifier
- Verifier verifies the signature, schema and checks that issuer and schema are registered at the Registry



W3C Verifiable Credentials

Verifiable Credentials are tamper-proof, digitally signed documents that securely prove claims about an individual or entity, allowing verification without directly involving the issuer.

- Specified by W3C - VC-Data-Model-2.0
- Defines a basic document structure and the essential features of the credential documents
- Can contain any kind of (json-representable) data
- Tamper-proof, due to digital signatures as part of the credential
- Already adopted by various regulatory and standardization bodys(f.e. EUDI Wallet Reference Architecture, OpenID4VC)

W3C Verifiable Credentials

```
{
        "@context": [
                "https://www.w3.org/2018/credentials/v1",
                "https://ec.europa.eu/eudi/v1"
        ],
        "type": ["VerifiableCredential", "NationalID"],
        "issuanceDate": "2024-01-01T12:00:00Z",
        "expirationDate": "2034-01-01T12:00:00Z",
        "issuer": "did:eu:country-code:issuer-identifier",
        "credentialSubject": {
                "id": "did:eu:country-code:user-identifier",
                "givenName": "Maria",
                "familyName": "Musterfrau",
                "dateOfBirth": "1983-12-01",
                "nationality": "AU"
        },
        "proof": {
                "type": "Ed25519Signature2020",
                "created": "2024-01-01T12:00:00Z",
                "proofPurpose": "assertionMethod",
                "verificationMethod": "did:eu:country-code:issuer-identifier#keys-1",
                "jws": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9..."
        }
}
```

Credential Metadata

Claim(s)

Proof(s)

VC - Metadata

```
{
        "@context": [
                "https://www.w3.org/2018/credentials/v1",
                "https://ec.europa.eu/eudi/v1"
        ],
        "type": ["VerifiableCredential", "NationalID"],
        "validFrom": "2024-01-01T12:00:00Z",
        "validUntil": "2034-01-01T12:00:00Z",
        "issuer": "did:eu:country-code:issuer-identifier",
        "id": "did:example:ebfeb1f712ebc6f1c276e12ec21"
}
```

- Provides contextual and technical information about the credential itself
- Helps the verifier to understand validity, provenance and usage context

- @context: Provides the context for interpreting the data within the credential, it's often used to link schemas and standards
- type: Defines the types of the credential to help verifiers interpret its purpose
- issuer: Information about the issuing entity. It can be used to verify the document.
- optional metadata like id, names, descriptions, validFrom, validUntil or credentialStatus supported

ficodes

VC - Claims

```
{
        "credentialSubject": [{
                "id": "did:eu:country-code:user-identifier",
                "givenName": "Maria",
                "familyName": "Musterfrau",
                "dateOfBirth": "1983-12-01",
                "nationality": "AU",
                "address": {
                        "street": "Kärtner Straße",
                        "city": "Wien",
                        "postalCode": "1010"

                }
        }
        {
                ...
        }]
}
```

- Section where the actual information or assertions are stored
- Property “credentialSubject” contains a set of objects, with the specific attributes or data pieces the issuer is attesting about the subject
- The claims usually should follow specific data schemas to ensure consistency and interoperability
- The subjects often contain id's, linking the claims to specific holders. It helps verifiers to confirm that a claim truly applies to the presenting subject.

VC - Proofs

```
{
        "proof": {
                "type": "Ed25519Signature2020",
                "created": "2024-01-01T12:00:00Z",
                "proofPurpose": "assertionMethod",
                "verificationMethod":
        "did:eu:country-code:issuer-identifier#keys-1",
                "jws":
        "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9..."
        }
}
```

- The proof contains details about the digital signature on the credential
- “type” contains the proof type. It determines what other fields are required to secure and verify the proof
- “proofPurpose” contains the reason the proof was created for.
- “verificationMethod” provides the information required to verify the proof
- “jws” contains the actual digital signature, which ensures the credential has not been tampered with

Verifiable Presentations

```
{
            "@context": [... ],
            "type": ["VerifiablePresentation"],
            "verifiableCredential": [
                      "@context": [ ... ],
                      "type": ["VerifiableCredential", "NationalID"],
                       "issuanceDate": "2024-01-01T12:00:00Z",
                       "expirationDate": "2034-01-01T12:00:00Z",
                       "issuer": "did:eu:country-code:issuer-identifier",
                       "credentialSubject": {
                                ...
                       },
                       "proof": {
                                ...
                      }],
           "proof": {
                      "type": "Ed25519Signature2020",
                      "created": "2024-01-01T12:00:00Z",
                      "proofPurpose": "assertionMethod",
                      "verificationMethod": "..-",
                      "jws": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9..."
           }

}
```

- A package of one or more Verifiable Credentials, created by the holder, to share specific information with a verifier
- Can combine multiple credentials, which is useful when a verifier required proof of several attributes
- The proof ensures integrity of the data and confirms that it was actually generated by the holder
- Enables the verifier to ensure a credential is provided by the actual holder, by checking the holder information in the credential with the presentations' proof

Verifiable Credentials - Formats

- Various types of Verifiable Credentials are specified at the moment:
  - jwt-vc: represents Verifiable Credentials as JSON Web Tokens. The format leverages established standards like JSON Web Signatures to ensure claims made in the credential can be validated be the verifier.
  - vc+sd-jwt: extension to the JWT standard, that enables privacy preserving VCs by allowing the holders to disclose only specific claims, while maintaining integrity and authenticity of the entire JWT.
  - ldp-vc: Linked Data Proof Credentials, use the LDP specifications
  - mdL: specialized structure for representing mobile drivers licenses

Decentralized Identifiers

Decentralized Identifiers(DIDs) are a new type of identifier that enables verifiable, self-sovereign digital identities without relying on a central authority, allowing individual to control their own identities and associated data securely and independent.

- Specified by W3C - Decentralized Identifiers
- Defines the syntax of an identifier
- Allows the controller of the identity to prove control without requiring permission from anyone else
- DIDs are URIs associated with a document allowing trustable interaction. The document can contain:
    - cryptographic material
    - verification methods
    - services to prove control

ficodes

Decentralized Identifiers

did:<method-name>:<method-specific-id><additional-components>

- did-prefix: indicates that the given URI is a decentralized identifier
- method-name: References a specific DID Method.
- The method specifies:
  - how the method-specific-id is generated
  - how the associated DID document is created
  - how the DID can be resolved to the document
  - how the DID document can be used to verify authenticity
- additional-components: reference into the DID-document or an external resource, depending on the method

ficodes

Decentralized Identifiers

- FIWARE Data Space Connector supports 3 methods:
  - did:key
  - did:web
  - did:elsi(in work)

did:key:zDnaeTTC781gfJKRjjShQV5sjNxXvgETXSctwfuMXZFmbeSE2

did:web:one.batterypass.fiware.dev:did

did:elsi:VATDE-309937516

did:key

did:key:zDnaeTTC781gfJKRjjShQV5sjNxXvgETXSctwfuMXZFmbeSE2

- a method that uses a public key as its identifier, directly incorporating cryptographic keys into the did
- fully self-contained, without any additional services required
- direct-key mapping allows straightforward verification of ownership
- easy to create: can be generated with standard key-creation mechanisms
- in the FIWARE DSC ideally used for identification of individuals or services(e.g. holders)

Decoding:

The id is the base58-btc encoded key, prefixed with the multicodec identifier of the key-type.

“zDn” as prefix of the ID indicates that it is a P-256 key.

did:web

did:web:one.batterypass.fiware.dev:did

- a method designed to be used in web-environments
- provides a way to create identifiers that are resolvable via standard web protocols
- the did-document is available at a well-known path, that can be created from the identifier
- in the FIWARE DSC, it could be used for participating organizations

Decoding:

Following the method-definition, the id can be resolved to the document address:

https://one.batterypass.fiware.dev/did/did.json

Https is enforced by the method and did as additional component provides information that the document is located under the sub-path /did.

did:elsi

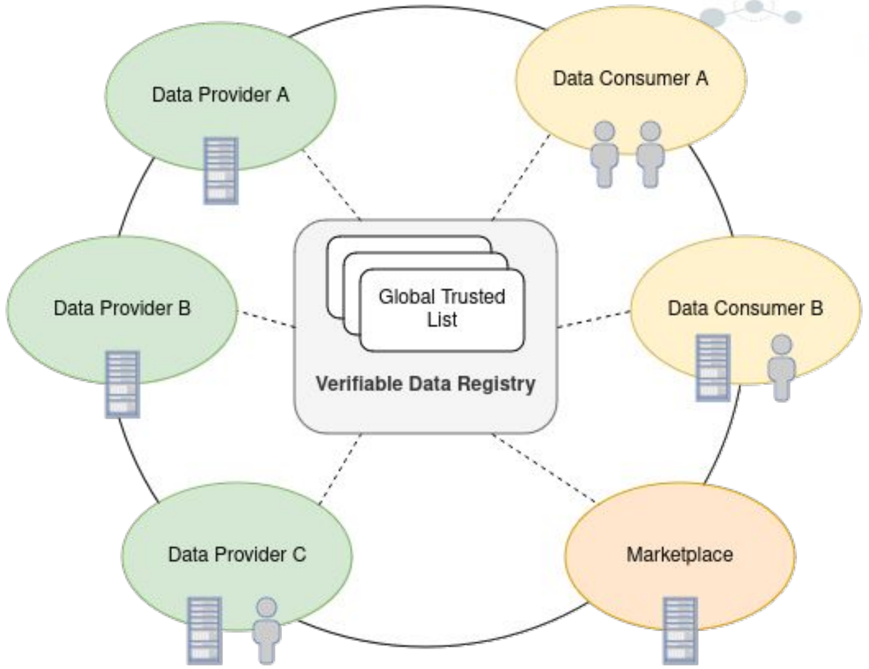
did:elsi:VATDE-309937516

- a method for legal persons, bridging the world of eIDAS regulation with Verifiable Credentials
- identifiers are derived from eIDAS certificates, issued to ETSI compliant legal persons. Examples would be:
  - VAT - based on the national tax identifier
  - NTR - based on national trade register
  - PDS - based on national authorization number of payment service providers
  - LEI - based on the global Legal Entity Identifier
- can be used to verify credentials following the JAdES specification, based on JWS
- according to the JAdES specification, certificate material is included in the JOSE Header and can be used for verification
- already used in the DOME-Marketplace

ficodes

Decentralized IAM in the FIWARE DSC

- Participants are registered in one/multiple Global Trusted Lists
- each participant can access the trusted list and check for registered participants
- additional Trusted Lists can be used, in order to incorporate public trust providers
- integration of GAIA-X Trust Framework could be achieved through this(more in session 6 of the webinar series)

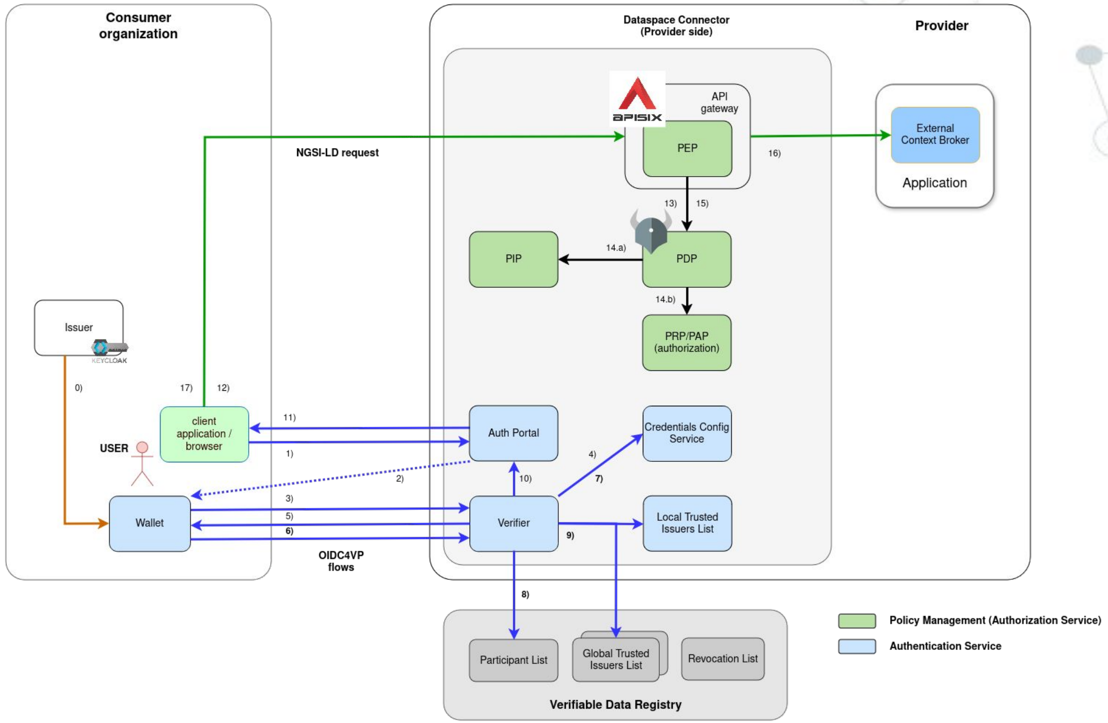


Decentralized IAM in the FIWARE DSC

0. Issuance:

- Consumer Organization issues Verifiable Credential to its user
- Follows the OID4VCI Standard
- Keycloak is used as the default issuer in a FIWARE Data Space
- Credential will be stored in a OID4VC compliant wallet

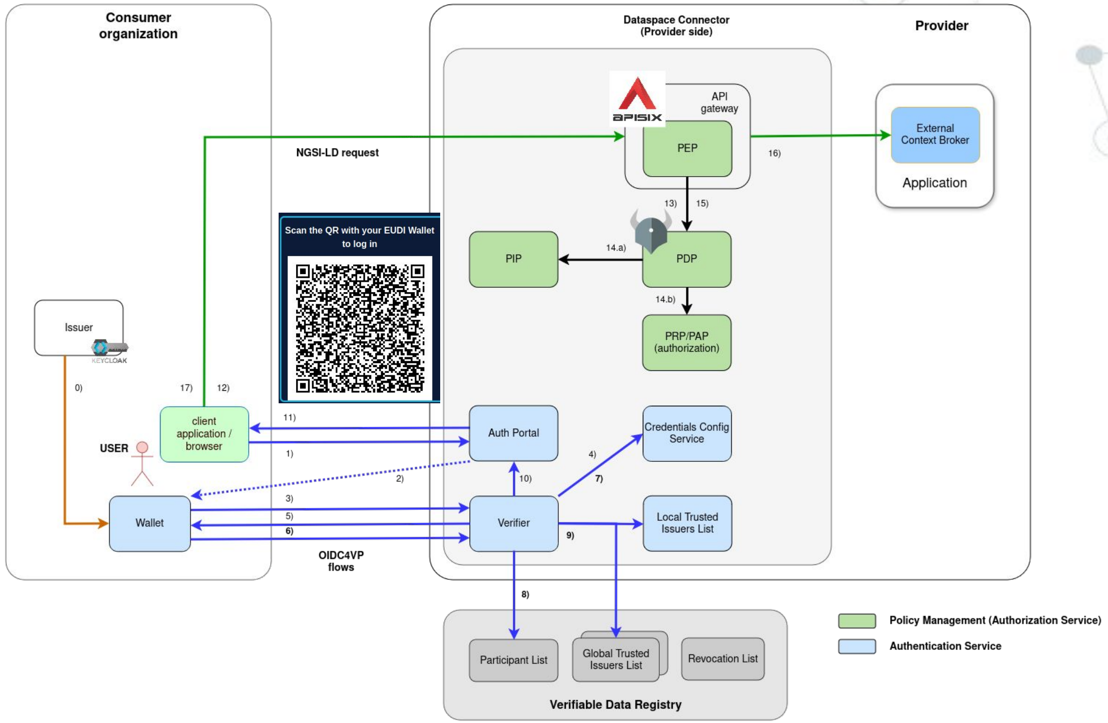
The user can now start its interaction with other participants, using the Credential to authenticate at them.



ficodes

Authentication

1. Users tries to access a protected resource and gets redirected to the Auth-Portal
2. Portal(as part of the Verifier) provides a QR with connection information for the Wallet
3. Wallet connects the Verifier with the provided information(f.e. service and scope) to start the OID4VP flow
4. VCVerifier requests the credentials configuration from the Credential Config Service
    - allow to define which credentials to be requested for a given service
    - defines which trusted lists to be used per credentials type/service



ficodes

Authentication - Credentials Config

The required Credentials for a given Service and Scope can be fine-grained configured through the Credentials Config Service.
It provides a REST-API to specify:

- the required Credentials
- the Trusted Lists to be used
- the supported Scopes and their configuration

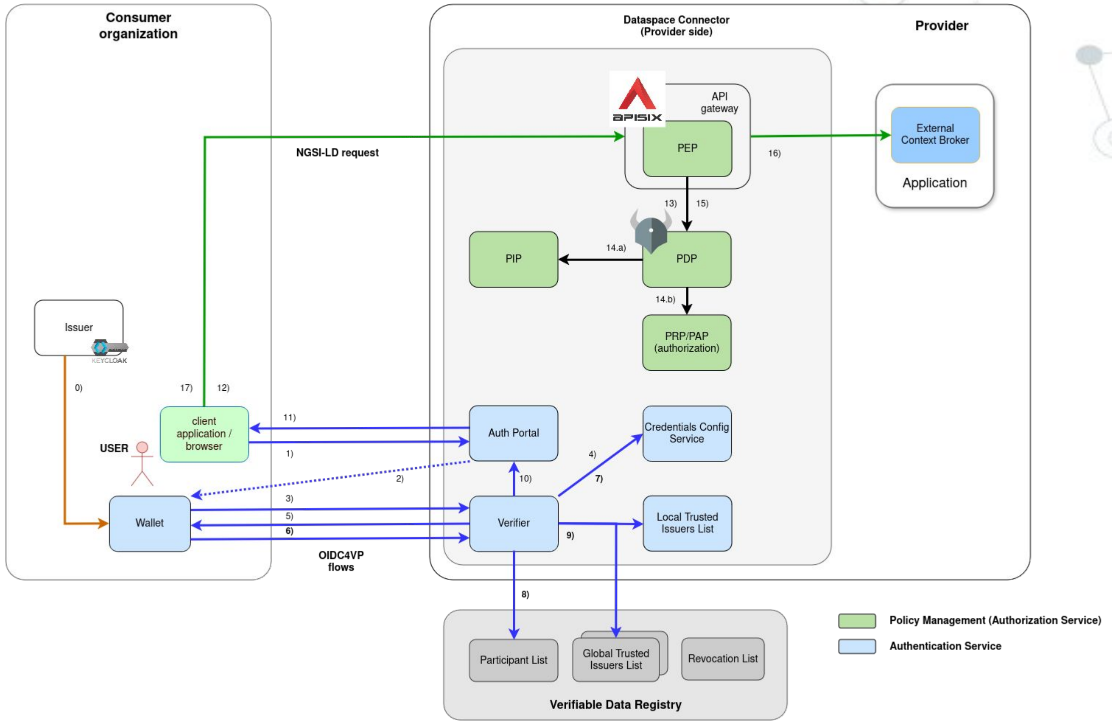
Example:

```
{
          "id":  "target-service",
          "defaultOidcScope": "read",
          "oidScopes": {
                    "read":  [{
                              "type":  "UserIdentityCredential",
                              "trustedParticipantsList": ["https://my-dataspace.org"],
                              "trustedIssuersList": ["https://my.default.svc.local"]
                    }],
                    "write":  [{
                              "type":  "EmployeeCredential",
                              "trustedParticipantsList": ["https://my-dataspace.org"],
                              "trustedIssuersList": ["https://my.default.svc.local"]
                    }],
          }
}
```

ficodes

Authentication

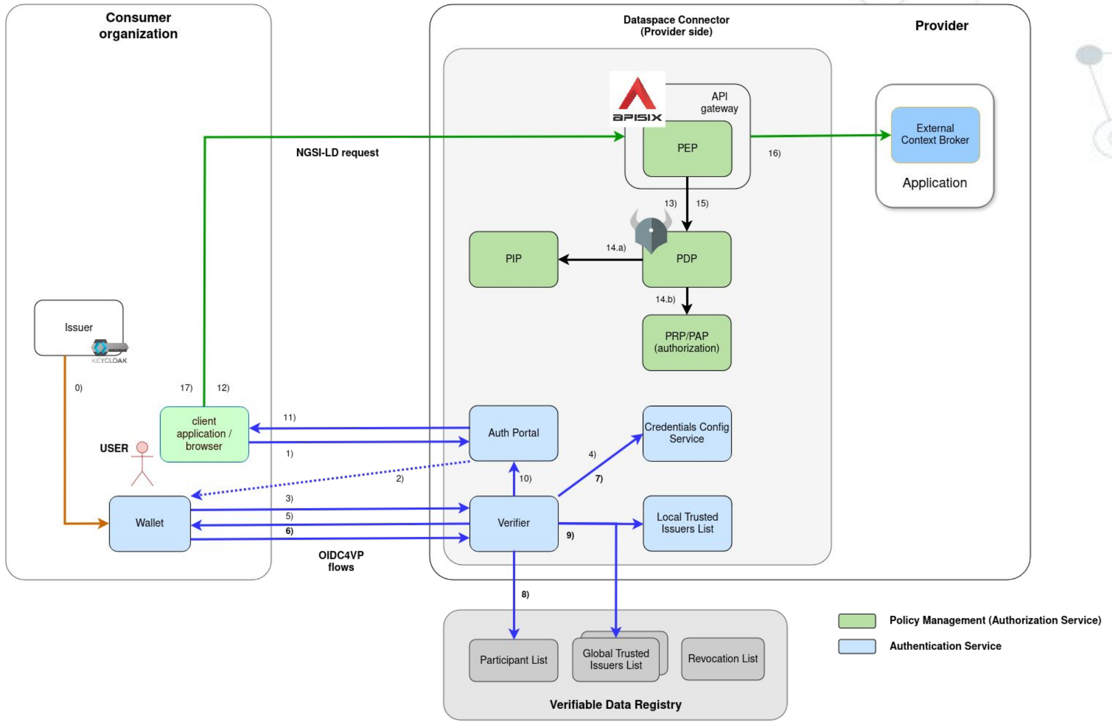
5. Verifier requests the credentials at the Wallet
6. Wallet creates the Verifiable Presentation and presents it to the Verifier
   - contains the requested Credentials
   - Signed with the Users Key
7. Verifier verifies the Signature of the Presentation and its contained credentials, checks at the Credentials Config Service if all Credentials are included and the Trusted Lists to be used.



ficodes

Authentication

8. Checks that the Credential was issued by a valid Data Space Participant
   - this is an optional step, that can be configured at the Credentials Config Service
9. Check the Credential at all configured internal or external Trusted Lists
   - the local Trusted Issuers List gives fine-grained control about the Claims to be trusted



Authentication - Trusted Issuers List

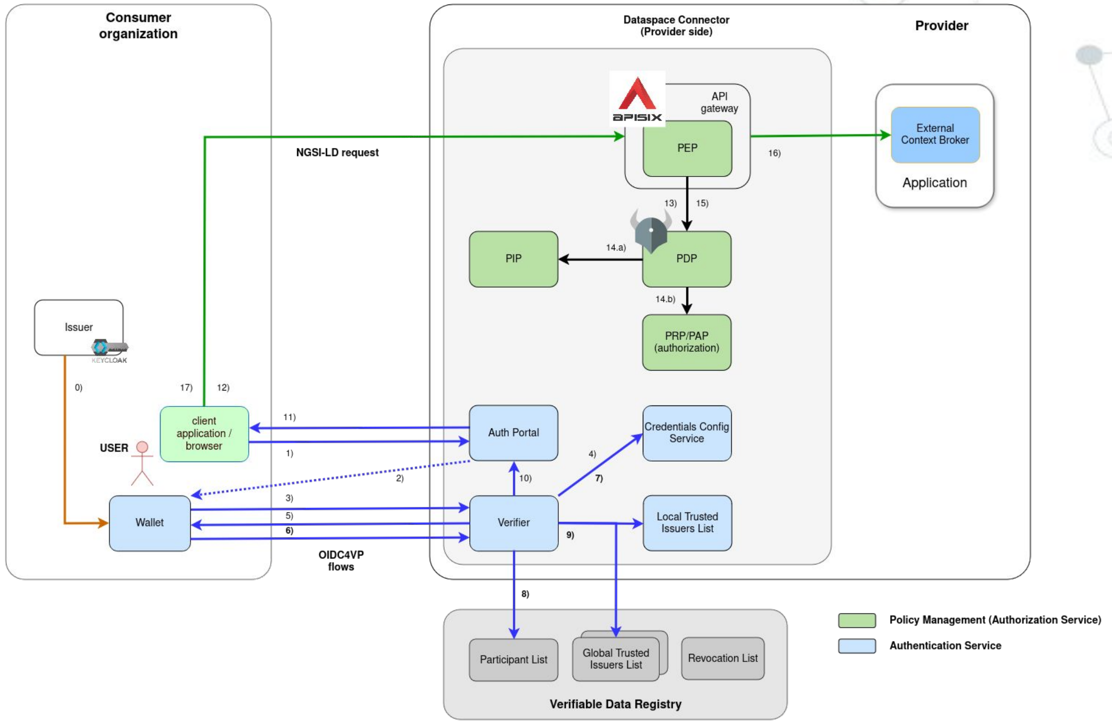
- implements the EBSI Trusted Issuers Registry API - FIWARE Trusted Issuers List
- in the ParticipantsList, only inclusion is checked
- IssuersList allows fine-grained control about trusted Credentials and Claims per Issuer
  - TimeRange that the configuration is valid
  - Types of Credentials the Issuer is trusted for
  - Claims and Values the Issuer is allowed to issue

Example:

```
{
        "did":  "did:web:my-consumer.org",
        "credentials": [{
                "validFor": {
                        "from":  "2024-12-21:T17:00:00Z",
                        "to":  "2025-12-21:T17:00:00Z"
                        },
                "credentialsType": "ConsumerCredential",
                "claims": [{
                        "name": "roles",
                        "allowedValues": [ "reader" ]
                }]
        }]
}
```

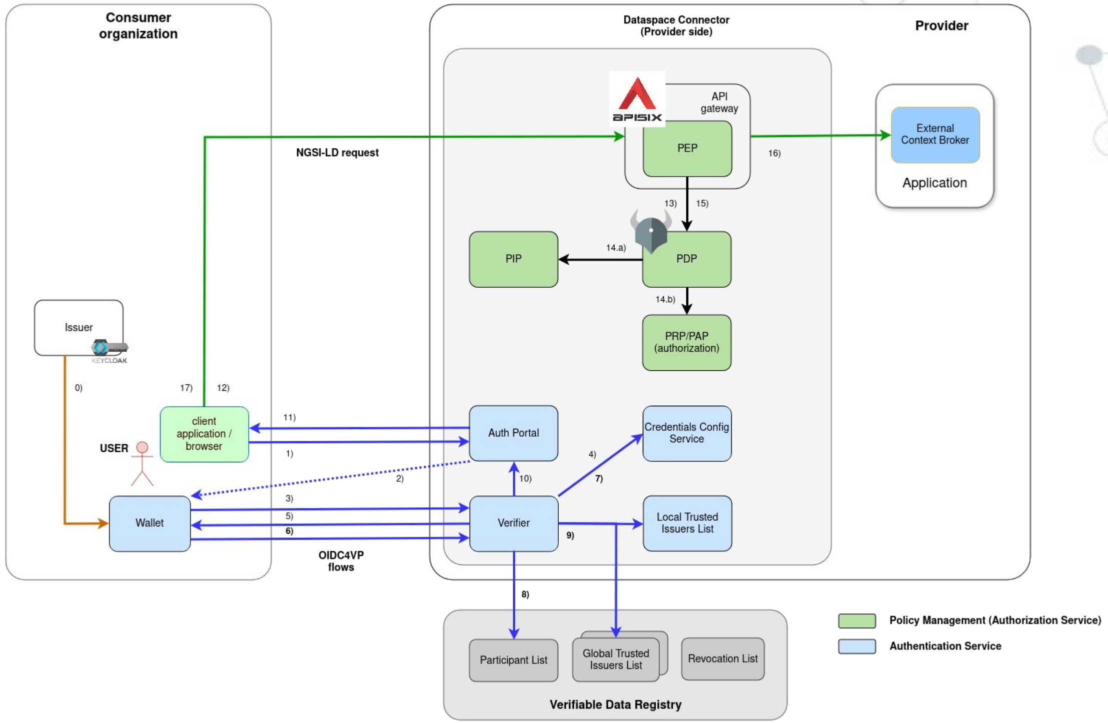
Authentication

10. If verification is successful, a JWT Token, containing the credentials information is provided to the frontend
    - the JWT is signed with a Key generated by the Verifier
    - Public Key is provided through a .well-known JWKS endpoint
    - downstream components only have to verify the JWT, instead of the full Verifiable Credential
    - easier integration with "old" OIDC-based services
11. JWT is provided to initiating application



Authorization

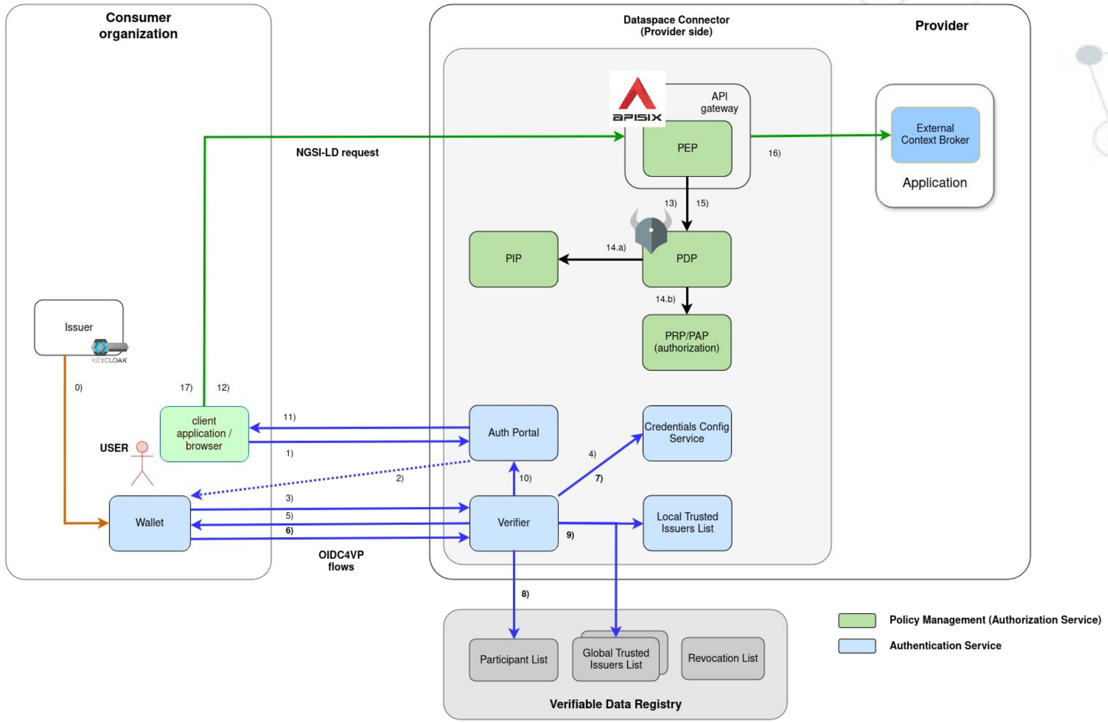
12. Application uses JWT to access the Provider Services through its Policy Enforcement Point(PEP)
    - APISIX is used as PEP
    - provides additional ApiGateway features
    - verifies the JWT at the Verifier and delegates the decisions to the Policy Decision Point(PDP)
13. PEP requests decision at the PDP
    - Open Policy Agent is used for policy evaluation
    - PEP provides request, request meta-data, authorization information and the request body



Authorization

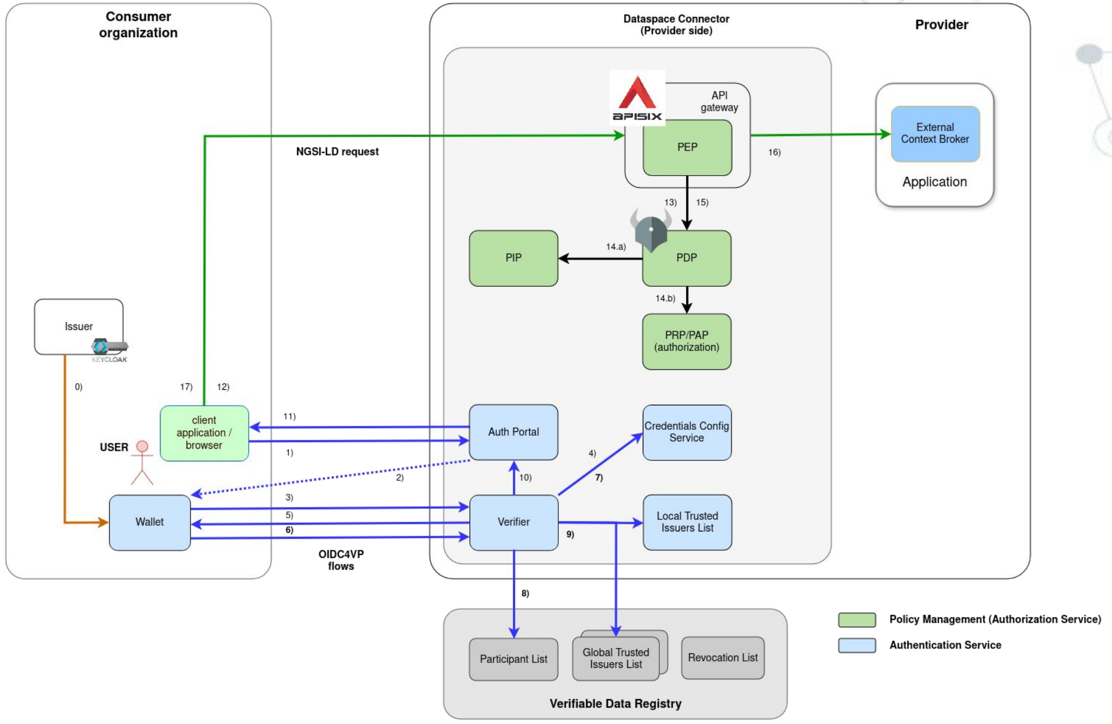
14. PDP evaluates the configured policies
    a. retrieves required information from the Policy Information Point(f.e. current time)
    b. retrieves actual policies to be evaluated(happens asynchronous, sequential integrated for better understanding)

- Open Policy Agent is used due to its speed, stability and established community
- already well integrated with APISIX(and various other ApiGateways)



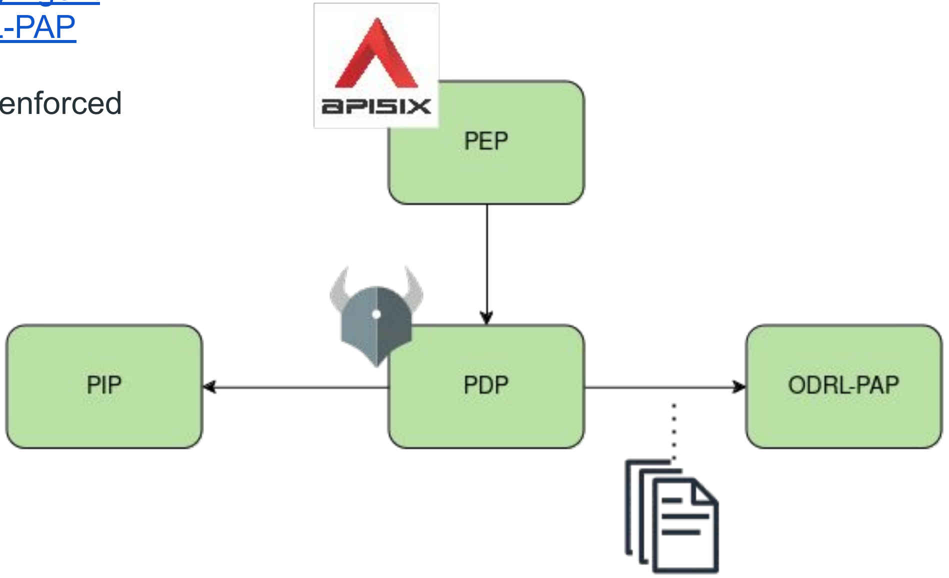
Authorization

15. PDP returns decision to the PEP
16. PEP forwards the request to the target service
17. Response is provided to the requesting application



Authorization

- handled by:
  - Policy Enforcement Point - APISIX
  - Policy Decision Point - Open Policy Agent
  - Policy Administration Point - ODRL-PAP
  - Policy Information Point
- Policies defined in ODRL, translated and enforced as Rego by OPA



ODRL

The Open Digital Rights Language(ODRL) is a standardized policy expression language used to represent permissions, prohibitions and obligations associated with digital content, enabling clear and enforceable rights management.

- Specified by W3C - ODRL
- allows to define policies on digital content
- Permissions can be used to express access-rights to resources
- possibility to couple them with resources - DOME ODRL Profile

ODRL

- @context provides the JSON-LD context for interpreting the policy structure
- @id is the unique identifier of the policy
- @type of the ODRL document, we only support odrl:Policy
- odrl:permission defines the actual rule:

“did:web:my-data-provider.org allows did:web:my-data-consumer.org to read the resource urn:ngsi-ld:entity:1"

```
{
          "@context":  {
                    "odrl": "http://www.w3.org/ns/odrl/2/"
          },
          "@id": "urn:uuid:6ee8b922-d09a-4621-8ba1-46b6e811f682",
          "@type": "odrl:Policy",
          "odrl:permission": {
                    "odrl:assigner": {
                              "@id": "did:web:my-data-provider.org"
                    },
                    "odrl:target": "urn:ngsi-ld:entity:1",
                    "odrl:assignee":  {
                              "@id": "did:web:my-data-consumer.org"
                    },
                    "odrl:action": {
                              "@id": "odrl:read"
                    }
          }
}
```

ODRL to Rego translation

- ODRL-PAP provides CRUD-Api for ODRL-Policies
- translates and stores policies
- policies are provided to OPA via the Bundles-API
- translation to Rego is configured through:
  - mapping.json connects policy entries with pre-defined Rego-Scripts
  - rego-folder contains all scripts provided for translation
  - can be extended and changed on deployment

Example:

"odrl:action": { "@id": "odrl:read" } is mapped to the rego-method "odrl_action.is_read(helper.http_part)"

- namespaces allow to define different behaviours for ODRL-Classes in the policies
- helper methods provide commonly used information, like http-headers, request-body or authorization information

Policy Enforcement

- every policy is condensed into a true/false decision
- all policies are combined to a single “main” policy
- Open Policy Agent evaluates the main-policy by individually evaluating the single policies
    - allows on first policy evaluated to “true”
    - rejects if no policy evaluates to true

Main Policy:

```
package policy.main

import rego.v1
import data.policy.cdsoxrwbuc as cdsoxrwbuc
import data.policy.vmixqkzkuu as vmixqkzkuu
import data.policy.paofakgsoc as paofakgsoc
import data.policy.gvitaqhecj as gvitaqhecj
default allow := false

allow if cdsoxrwbuc.is_allowed
allow if vmixqkzkuu.is_allowed
allow if paofakgsoc.is_allowed
allow if gvitaqhecj.is_allowed
```

Examples

- restrict accessible targets - only entities of type “EnergyReport”

```
{
	"@context":  {...},
	"@id": "urn:uuid:6ee8b922-d09a-4621-8ba1-46b6e811f682",
	"@type": "odrl:Policy",
	"odrl:permission": {
		"odrl:assigner": "did:web:my-data-provider.org",
		"odrl:target": {
			"@type": "odrl:AssetCollection",
			"odrl:source": "urn:asset",
			"odrl:refinement": [{
					"@type": "odrl:Constraint",
					"odrl:leftOperand": "ngsi-ld:entityType",
					"odrl:operator": "odrl:eq",
					"odrl:rightOperand": "EnergyReport"
				}]},
		"odrl:assignee":  "vc:any" ,
		"odrl:action": {
			"@id": "odrl:read"
		}
	}
}
```

ficodes

Examples

- restrict assignees - only assignees with the Role “Operator” and the credential type “OperatorCredential”

```
{
          ….,
          "odrl:permission": {…
                    "odrl:assignee":  {
                              "@type": "odrl:PartyCollection",
                              "odrl:source": "urn:user",
                              "odrl:refinement: {
                                        "@type": "odrl:LogicalConstraint",
                                        "odrl:and": [{
                                                            "@type": "odrl:Constraint",
                                                            "odrl:leftOperand": "vc:role",
                                                            "odrl:operator": "odrl:hasPart",
                                                            "odrl:rightOperand": {
                                                                      "@value":  "OPERATOR",
                                                                      "@type":  "xsd:string"}},
                                                  {
                                                            "@type": "odrl:Constraint",
                                                            "odrl:leftOperand": "vc:type",
                                                            "odrl:operator": "odrl:hasPart",
                                                            "odrl:rightOperand": {
                                                                      "@value":  "OperatorCredential",
                                                                      "@type":  "xsd:string"}
                                                  }]}}
                              }
}
```

ficodes

Examples

- use(e.g. CRUD) is allowed between 31.12.2023 and 31.12.2024

```
{
          ....,
          "odrl:permission": {
                    "odrl:target":  "urn:ngsi-ld:data-entity:1",
                    "odrl:assignee":  "did:web:my-data-consumer.org",
                    "odrl:constraint":  [{
                                        "@type": "odrl:Constraint",
                                        "odrl:leftOperand": "odrl:dateTime",
                                        "odrl:operator": "odrl:gt",
                                        "odrl:rightOperand": {
                                                  "@value":  "2023-12-31",
                                                  "@type":  "xsd:date"
                              },
                              {
                                        "@type": "odrl:Constraint",
                                        "odrl:leftOperand": "odrl:dateTime",
                                        "odrl:operator": "odrl:lt",
                                        "odrl:rightOperand": {
                                                  "@value":  "2024-12-31",
                                                  "@type":  "xsd:date"
                              }],
                    "odrl:action":  "odrl:use"
}
```

Examples

- use external data for evaluation

```
{....,
        "odrl:permission": {
                "odrl:target":  {...
                        "@type": "odrl:AssetCollection",
                        "odrl:refinement": {
                                "@type": "odrl:Constraint",
                                "odrl:leftOperand": "dome-op:owner",
                                "odrl:operator": "odrl:eq",
                                "odrl:rightOperand": "dome-op:currentParty",
                                "odrl:dataType": "xsd:anyURI"
                        }
}

related party(http part) := rp if {
   path without query := split(http part.path, "?")[0]
   ## will be one or multiple entities
   responseBody := http.send({"method": "get", "url": sprintf("%v%v", [http_part.host,
path without query])}).body
   rp = responseBody.relatedParty
}
owner(related party) := o id if {
   owner rp := [rp | some rp in related_party; rp.role = "Owner"]
   o_id = owner_rp[_].id
}
```

ficodes

Links

- Slides:
  - https://github.com/wistefan/presentations
- FIWARE Data Space Connector:
  - https://github.com/FIWARE/data-space-connector
- ODRL-PAP:
  - https://github.com/wistefan/odrl-pap
- DOME-Marketplace:
  - https://dome-marketplace.org/dashboard

ficodes

contact@ficodes.com | www.ficodes.com | +34 614 20 74 47 | C/ Hespérides 5 (28232) | Las Rozas de Madrid