PrivacyCred

Release 0.9.2

Jesus Ruiz

Mar 29, 2021

CONTENTS

PrivacyConsent is a system that allows a citizen to manage informed consent to two independent entities to exchange personal data among them for a very specific purpose. PrivacyConsent is based on PrivacyCred, a generic credential system which is designed to be secure, privacy-preserving, scalable, performant and robust.

PrivacyCred is designed specifically for some important use cases where especially sensitive personal data is handled, like when people at risk of exclusion is involved or with some health data flows. A high degree of privacy and unlinkability is the first design criteria. It also supports scenarios where physical, on-person verification of identity of holder is needed and where normal W3C Verifiable Credential flows are not fully suitable as they are normally designed currently.

PrivacyConsent

The first part describes the *PrivacyConsent* flows on top of PrivacyCred for very strict use cases, like managing the informed consent of a user in the collective of people at risk of exclusion, when two or more entities are involved in the secure and legal exchange of sensitive personal data, to collaborate to provide a better service to the citizen.

PrivacyCred

The secod part describes the *PrivacyCred* system, including the data model, interfaces and main interactions. Understanding the caharacteristics of the underlying system is important to understand how PrivacyConsent works.

CHAPTER

ONE

PRIVACYCONSENT

Once the PrivacyCred system is operating and the relevant legal entities are registered in the Trust Framework, the management of PrivacyConsent credentials is relatively easy, as it can be implemented using otherwise standard credentials using the underlying PrivateCred system, and benefiting from its specific privacy characteristics.

1.1 Credential issuance by first entity

The flow of issuance of a PrivacyConsent credential is mostly the same as any other credential. However, it includes some special items. The overall flow is:

- The citizen logs into the system of the Issuer Entity. If the entity is a private company, we assume that the citizen is already an existing customer and that the proper KYC processes have benn performed when the customer was onboarded. This process can also be performed over the phone if the business has a proper mechanism of identifying the customer in order to perform legaly binding transactions. If the entity is a public administration, we assume that the entity performs the identification with the nationally acceptable mecanisms, including in-person and remote mechanisms.

  This standard login process has to be performed at least once to have the legal backing of credentials issued in the following steps. After the credential issuance, the Issuer entity could use (if desired) the credential to perform automatic and remote login of the citizen.

- The citizen explicitly accepts to provide consent for a given purpose. The operation should be performed in the same way as if the customer were executing other legaly binding operations in the Issuer Entity system. This is acting effectively as a signature of the fact that the citizen is providing consent for a given purpose, including the consent to generate a digital vertifiable credential to facilitate some process.

- The Issuer Entity records the consent for future reference or auditing, as it is done currently without verifiable credentials.

- The Issuer entity then requests from the Citizen a new credential that should include a Peer DID that the Citizen generates with her mobile app. The credential should also be signed with the private key associated to that Peer DID. There may be several mechanisms for requesting and sending credentials. The PrivacyConsent system is based on PrivacyCred so it uses the same set of mechanisms, including QRs.

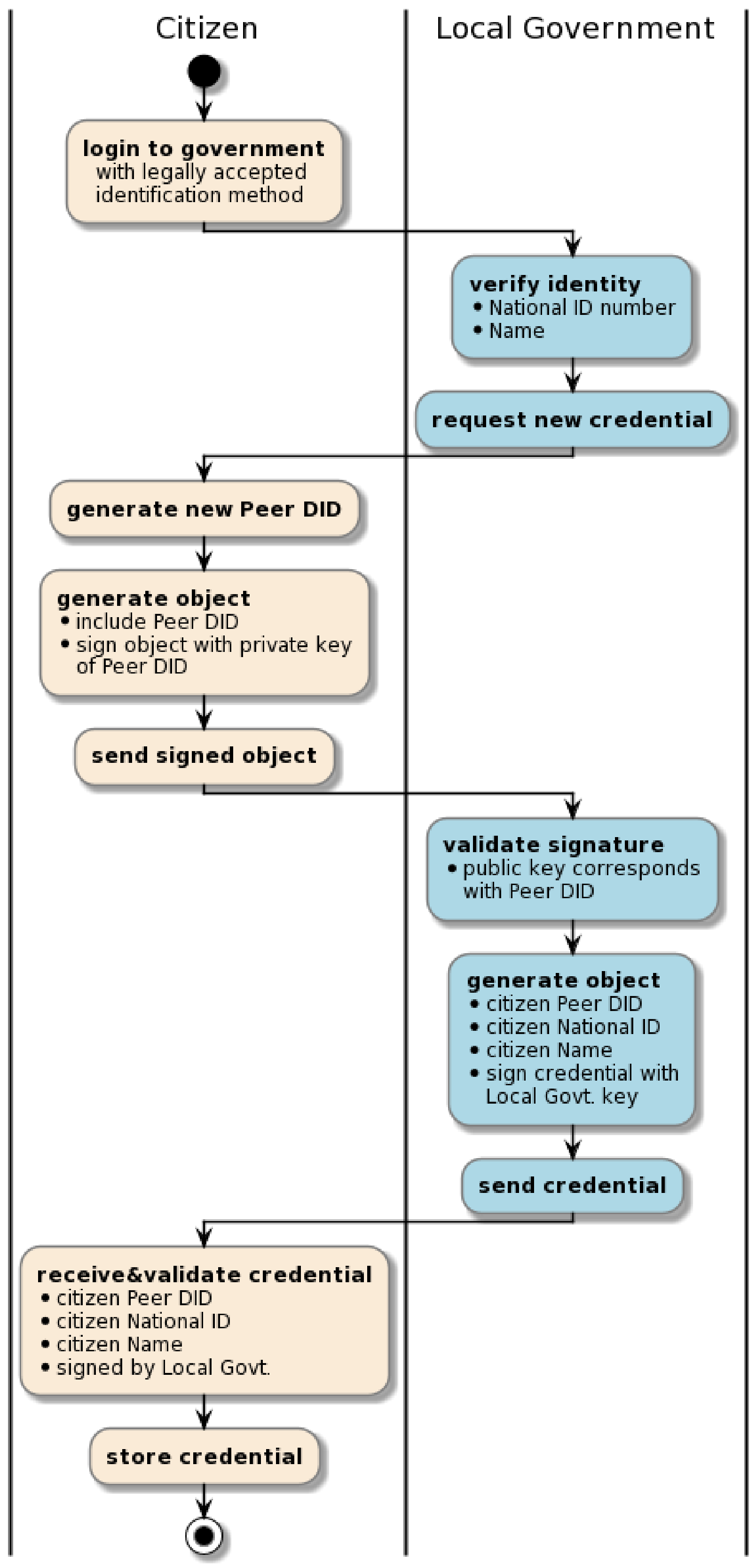


Fig. 1: Credential issuance by first entity

- The Issuer entity receives the credential from the Citizen and performs some validations.
- The Issuer Entity generates a Verifiable Credential that includes the explicit consent and purpose, and the citizen Peer DID. The credential is digitally signed by the Issuer Entity with the private key associated to the public key registered in the Trust Framework. Including the Peer DID of the citizen, the credential is binding that Peer DID with the real identity of the citizen (as verified using the KYC data). And this binding is digitally signed by the Issuer Entity in a tamper-resistant way.
- The citizen receives the credential via the QR mechanism (or any other that is supported by the system).

After the interaction, both parties (citizen and Issuer Entity) have a digital representation of the informed consent:

- The Issuer Entity has a consent with a proof of citizen acceptance based on the KYC process that was performed when onboarding the citizen.
- The citizen has a credential digitally signed by the Issuer Entity attesting that the citizen provided consent and that it was accepted by the Issuer Entity.

1.2 Credential reception by second entity

After the previous process, the first entity has already the consent from the citizen. The process to send the consent to the second participating entity is based on the verification flows. The process is the following:

The citizen interacts with the Verifier Entity. This could be done via different channels, like login into the system of the Verifier Entity, or in-person in the premises of the entity. It can also be performed via other electronic channels like email or even messaging systems. Here we describe the process for in-person interaction as when the citizen goes to the social services for the first time and she does not have yet any legally accepted identification mechanism to interact with the public administration. After the first interaction, all other interactions could be performed remotely.

The citizen uses her mobile app to generate a W3C Presentation object, which is essentially a digitally signed object including one or more credentials. The Presentation object is digitally signed with the private key associated to the Peer DID that was used when the process of generating the credential was performed by the first entity.

In this way, the Presentation object includes the following:

- The purpose of the consent.
- The proof that the Issuer Entity is attesting that it received the explicit consent from the customer, after identifying her.
- The signature of the whole thing using a private key associated to the Peer DID that the citizen provided to the Issuer Entity.

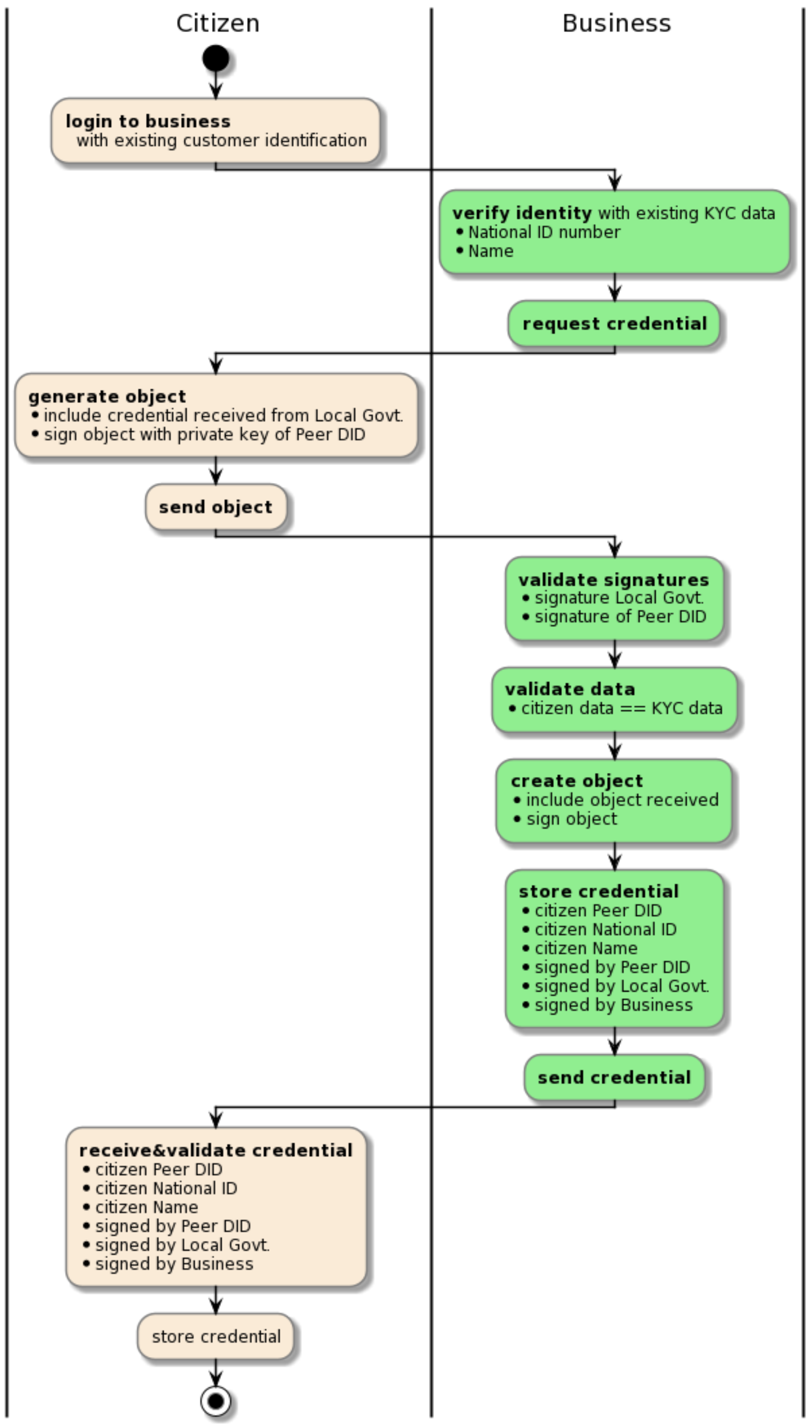


Fig. 2: Credential reception by second entity

CHAPTER

TWO

PRIVACYCRED

2.1 Requirements

The journey of a certificate is completed in 3 distinct steps:

1. the collection and registration of data about the citizen and her associated characteristics,
2. the issuance of certificate, and
3. the presentation of the certificate to a verifier for its verification.

A certificate should rely on a minimum dataset. Included in the minimum dataset is a Unique Certificate Identifier (UCI), which could be used as a link to the underlying data registry.

The verifier of a certificate should be able to establish that:

- The certificate has been issued by an authorised entity;
- The information presented on the certificate is authentic, valid, and has not been altered;
- The certificate can be linked to the holder of the certificate;

2.1.1 Main design principles and business requirements

The design of the trust framework for interoperable issuing of certificates and verification of their integrity and authenticity relies on key design principles listed below. The list is not prioritised. Instead, the trust framework that is specified later in the document attempts to optimise as many of the key design principles as possible.

Data protection (including data minimisation, purpose limitation, etc.) The trust framework should protect the data of the involved individual stakeholders (most importantly, certificate holders). This covers several data protection dimensions catered by the General Data Protection Regulation, including purpose limitation and data minimisation. In practice, only the bare minimum set of data that is required for the supported use cases should be processed (data minimisation) and the purpose of data collection should be checked against the use cases (purpose limitation).

Similarly, only the bare minimum set of data that is required for the supported use cases should be presented to a specific verifier (data minimisation) and the purpose of data presentation should be checked against the use cases (purpose limitation).

In order to achieve the latter, the trust framework may support different presentation datasets for different verifier scenarios. The data protection principle has a strong impact on the specification of the Minimum Dataset and the design of the use cases of the trust framework.

Data security and privacy by design and by default Abuse of data by actors (especially, the certificate verifiers and holders) and forgery should be prevented by any reasonable means. The trust framework should by design and default ensure the security and the privacy of data in the compliant implementations, ensuring both security and privacy.

Available tools should be used for restricting access to data and preventing malicious use of data, while the establishing of the authenticity of data and its link to the certificate holder should be ensured. The design should prevent the collection of identifiers or other similar data which might be crossreferenced with other data and re-used for tracking ('Unlinkability').

Inclusiveness (especially medium-neutrality) The trust framework should be inclusive especially towards the individual citizen ('no citizen left behind').

The design of the trust framework should attempt to maximize its support for diverse contexts (e.g., high resource vs low resource contexts).

To enable this, the trust framework should support a spectrum of certificate presentation media from plain paper certificate to augmented paper certificates (e.g., paper certificate with printed machinereadable parts such as barcodes, QR codes, Machine Readable Zones) and to purely digital certificates (e.g., in-app certificates).

The PrivacyCred system supports paper certificates, augmented paper certificates, QR codes and purely digital certificates.

In addition, contrary to many SSI Verifiable Credentials implementations, the PrivacyCred system is implemented as a PWA (Progressive Web App) that can be used simply in a mobile browser (or tablet/PC) without installing anything in the device or registering for anything.

However, the user can install the PWA in the device if the user so wishes to facilitate future uses. In any case, the system does not require any type of registration of the identity of the user.

Simplicity and user-friendliness It is very important that the trust framework is designed with simplicity and user-friendliness of the possible implementation of digital certificate systems in mind.

More formally, the trust framework should not have features or functionalities that would unnecessarily complicate the resulting implementation of a digital certificate system or make them unnecessarily difficult to use. Lack of simplicity could increase the time it takes to implement the compliant digital certificate systems, while lack of user friendliness could hinder the uptake of the resulting implementations. User-friendliness is relevant for quick and easy processing, specifically to certificate holders and to verifiers.

The PrivacyCred system follows the rule of **Occam's Razor** eliminating any feature or functionality which is not strictly required for the use case.

This not only provides simplicity and user-friendlyness but also provides a system easier to understand and maintain which is more secure and robust.

Implementation flexibility The trust framework specifications should provide implementers with a variety of options when developing digital certificate systems according to the trust framework specifications. This key design principle aims at reducing the implementation time and leveraging/reusing existing infrastructures in involved entities.

To satisfy this principle, the trust framework specifies, whenever possible, a list of alternative methods, flows, architectures and implementation options, for example alternative presentation media, verification options, implementation technologies, etc. whilst still guaranteeing the same level of trustworthiness.

Modularity and scalability This is strongly linked with the previous key design principle. The trust framework architecture should be modular and easily scalable, for instance, to additional usage scenarios, use cases and types of certificates.

The trust framework already supports different usage scenarios (e.g. alternative settings in which certificates may be requested or verification may take place).

Open standards The trust framework should rely for its implementations on open standards, to the extent that this is possible. This will greatly contribute to the interoperability of the resulting implementations, in addition combined with open governance and open source implementations, it will instil trust in the involved stakeholders.

Cross-border interoperability Implementations of certificates that comply with the specifications of the trust framework should be interoperable, and not only at the national level.

This means that if Countries A and B implement the specifications, it should be possible for a verifier in Country B to verify a digital vaccination certificate that has been issued in Country A.

Cross-border interoperability should be ensured across EU and EEA countries. The Trust Framework should not prevent interoperability with the solutions designed on a global level.

2.1.2 ID binding and verification

An important parameter of the trust framework pertains to the identity of the subject of the certificate i.e., the person for whom the certificate is issued. The identity of this subject shall be bound to a certificate when the latter is issued (ID binding) and has to be verified when the certificate is being presented and verified (ID verification). These two processes (ID binding at the Issuance step and ID verification at the Presentation and Verification step) prevent possible impersonation attempts (i.e., a person fraudulently presenting a certificate that has been issued to someone else as if it were their own), and are in line with the data security and privacy by design and default principles of the trust framework.

The processes of ID binding and/or verification shall rely on (nationally and/or internationally) established methods for ID binding and verification. In other words, the trust framework does not specify in its architecture dedicated components or modalities for undertaking the ID binding and verification process.

The recommended methods for performing ID binding and verification are based on nationally issued identity proof documents, such as national IDs and passports, and regulated customer onboarding processes (in the case of private companies). The binding is performed at the time of issuance (ID binding) and verification (ID verification) of the certificate and therein personally identifying information held in the syetems of the entities involved should be compared against the information in the certificate.

Contrary to many SSI Verifiable Credentials implementations, the PrivacyCred system does not require any registration on the part of the user like registering her DID in the blockchain or any other repository, as the system relies in pre-existing identification processes (e.g., KYC for private companies).

The only personal information managed by the system is the one in the minimum dataset as specified in this document. The personal data elements are incorporated to the certificate and not used for any other thing or purpose. It is assumed that the minimal person identification data specified in this document can be used to perform the ID binding with a national ID, passport or any other suitable nationally issued identity document.

2.2 PrivacyCred: General description of the system

2.2.1 Main components

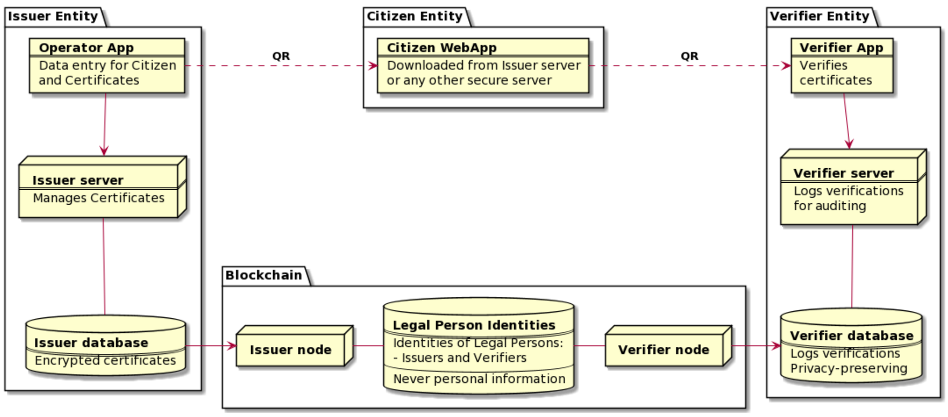


Fig. 1: Components of the system.

The main components are the following:

Issuer Entity The juridical person that digitally signs and issues a credential to the User. The Issuer Entity employs or subcontracts the actual people performing the process called Issuer Operator in the diagram. The Issuer Entity assumes full responsibility for the legal implications of the issueance process, especially GDPR compliance. The Issuer Entity acts as a Data Controller with respect to the Personal Information collected from the Citizen when the certification is issued.

Issuer Operator The natural person that is employed/subcontracted by the Issuer Entity to actually drive the process of issuing the credential on behalf of the Issuer Entity.

Issuer Operator App This is the application used by the natural person that drives the issuance of the credential. The application allows the operator to enter the details of the user and of the credential and issues the credential to the user on behalf of the Issuer Entity. It is the responsibility of the Issuer Entity to ensure that the Operator performs the process in the right way.

Citizen This is the natural person that receives a credential and may present it when needed.

Citizen WebApp This is the application used by the end user to manage the credentials. The reference implementation is not a native application but rather a PWA (Progressive Web App), which can be used either as a normal web app (without installation) or it can be installed and used in a very similar way to a native mobile app. The characteristics of this app are explained later.

Verifier Entity A juridical person that verifies the credential. In the process of verification, the Verifier Entity receives personal data from the Citizen. The Verifier Entity is responsible for compliance of all applicable regulations, including GDPR.

Verifier Operator A natural person that verifies the credential. It is important to distinguish between natural and juridical persons in the verification process because the flows may be different as the regulatory implications may be different. The diagram does not explicitly mention the Verifier Person, but it will be described in detail later in the document.

Verifier App The application used to verify the credential presented by the user. The reference application can be used either by an employee of a Verifier Entity or by an individual natural person, as explained later.

Blockchain This should be a Public-Permissioned blockchain network as a general-purpose blockchain network which is used to implement the Trust Framework allowing the efficient and secure verification of credentials. It is never used to store personal information. Personal information management is the responsibility of the legal entities Issuer Entity and Verifier Entity, and they are responsible for compliance to applicable regulations, especially GDPR. There may be more than one blockchain network, and the system is very interoperable across networks. The specific interoperability features are describer in a specific section later in this document.

2.2.2 Main credential flow

1) **Verification of User and Credential issuance**

The Issuer Operator identifies the User (in the same way as an airline employee identifies passengers before boarding a plane) and uses her mobile app to enter the details of the User. In the initial implementation of the system the operator has also to enter manually the details of the Credential to be issued. It is the responsibility of the Issuer Operator (and ultimately of the Issuer Entity) to ensure the veracity of both the User details and the Credential details. This is a critical point in the system, as the level of trust in the credentials will depend on the level of trust of the issuance process.

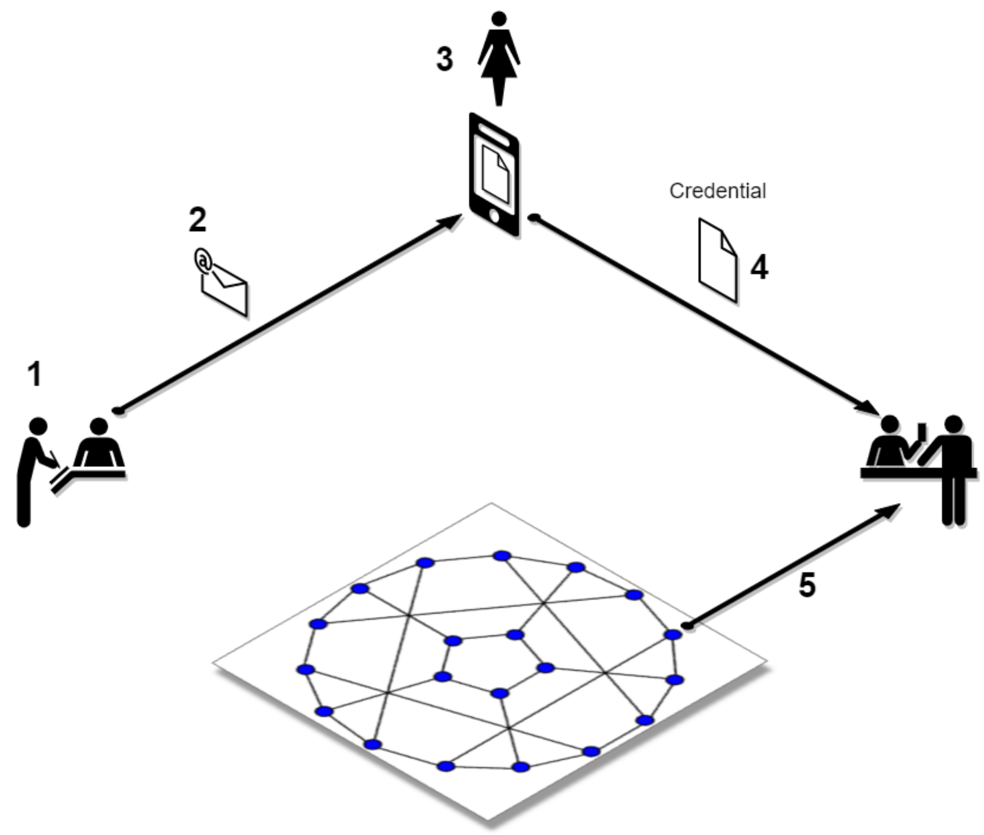


Fig. 2: Main credential flow.

2) **Sending the Credential to the Citizen**

   The Credential is sent to the User. There are several possible flows, using different channels (email, QR, etc.). The main one is using QR codes and is the following:

   1. The Issuer Operator displays the credential for the User in her mobile phone screen, in a QR format. More detaisl about the specific QR format later.
   2. The User scans the QR using her mobile web app.
   3. The mobile web app of the User gets the Credential and stores it in the storage of the mobile device.

3) **Store the Credential**

   The Credential is stored in the mobile phone of the User. In the reference implementation it is stored in the IndexedDB local database. More than one credential can be stored in the mobile. A Citizen could for example store credentials of other persons of the family when traveling, or a history track of credentials received during a vacation. More details are given later in this document.

4) **Present the Credential**

   When the Citizen has to prove something, she sends the Credential to the Verifier. As before, there are several possible flows, the main one using QR codes:

   1. The User display the Credential in her mobile phone in QR format.

2. The Verifier scans the QR from the User mobile screen
3. The mobile app from the Verifier receives the Credential and verifies it.

5) **Verify the Credential using the Trust Framework in the blockchain**

   The Verifier mobile app verifies formally the Credential with the signature, and then checks that the signature of the Credential corresponds to an authorized Issuer Entity registered in the Trust Framework in the blockchain. The verification process is essentially the one described in the W3C VC specifications.

2.3 The Trust Framework: bootstrapping the system

Before the issuance of credentials can take place, the system has to be bootstrapped and setup. There are two processes that have to be performed:

1. A One-time process at the beginning of the whole system: involves things like deploying Smart Contracts and initializing them with the parameters of the system.
2. A process for the onboarding of each new Issuer Entity and Verifier Entity. This process is basically generating and registering in the blockchain the Identity of the entity entering the system.

2.3.1 Public-Permissioned blockchain network

The system requires at least one Public-Permissioned[1] blockchain network. The network should be trusted, efficient, publicly available and compliant with all applicable regulations.

The system is designed to be easily interoperable with other Public-Permissioned blockchain networks, like LACChain or EBSI. This is described in detail in the appropriate section of this document.

2.3.2 Information in the blockchain and Personal Identifiable Information (PII)

No personal information is ever recorded on the blockchain. The blockchain is only used to register the identities of the legal persons involved in the system. The information recorded for businesses and organizations includes:

- Public identification information of the legal person in the current regulatory environment, like VAT number, LEI (Legal Entity Identifier[2]), or any legally accepted identification in the countries implementing the the system.
- Some commercial information, like the web site
- The public key used to verify the Verifiable Credentials digitally signed by the legal entity

[1] https://github.com/hesusruiz/PublicPermissionedBlockchain
[2] https://www.gleif.org

The diagram below shows the registration of a new Issuer Entity in the blockchain. There are two types of legal persons registered in the blockchain:

1. **Issuer Entity**: a legal person has to be properly registered before it can issue any credential that can be verified by other actors in the system.
2. **Verifier Entity**: a legal person that receives and verifies credentials from natural persons has to be registered in the blockchain. When the legal person receives the credential (which includes personal data), this fact is registered in order to enhance auditability of the system later. This registration is performed in a privacy-preserving and scalable way. The process is described in detail later in this document. Natural persons can also verify credentials, but the verification process is different in order to avoid pre-registration of natural persons. This is described in detail later.

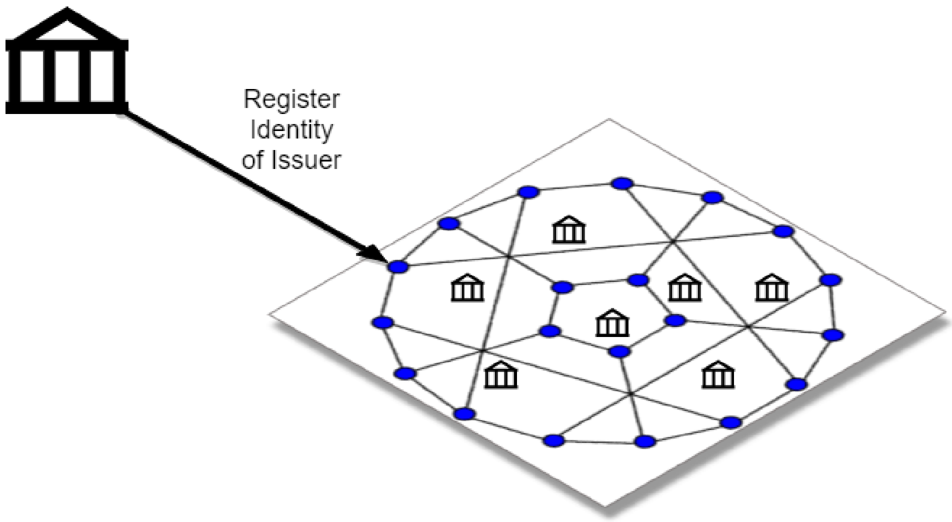


Fig. 3: Trusted Registry of Legal Entities in the blockchain.

2.3.3 Trust Framework: trusted registration process of legal entities

The trust framework is designed to be largely decentralised.

The identities of the legal persons involved in the ecosystem are registered in a common directory implemented in the blockchain following a hierarchical scheme very similar to the DNS (Domain Name Service) schema in the Internet. Once an entity is registered in the system, it is completely autonomous for adding other entities that are managed as child entities.

However, there is one centralised element: the root of trust at the top of the hierarchy should be a trusted entity in the ecosystem that is the one bootstraping the system. Typically it should be a regulatory body or a public administration.

The approach is described in the following figure.

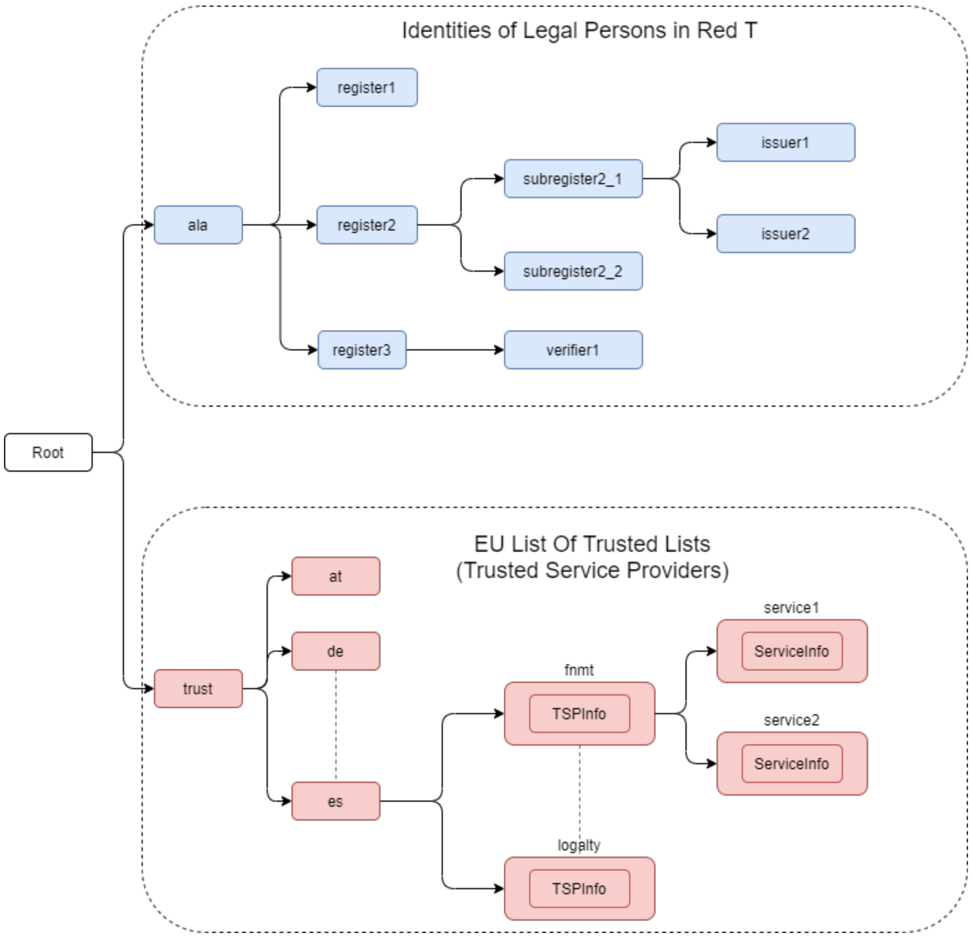


Fig. 4: The Trust Framework in the blockchain.

Creating identities

A new identity can only be registered as a sub-node by an existing entity already registered in the system. The API used is `/api/did/v1/identifiers` and its definition is the following:

`POST /api/did/v1/identifiers`

Create an Identity anchored in the blockchain.

Request JSON Object

- **`DID`** (*`string`*) – the DID of the new identity, example: "did:elsi:VATES-B60645900"
- **`domain_name`** (*`string`*) – Domain name to assign in the hierarchy, example: "in2.ala"
- **`website`** (*`string`*) – Website of the entity, example: "www.in2.es"
- **`commercial_name`** (*`string`*) – Commercial name, example: "IN2 Innovating 2gether"
- **`new_privatekey`** (*`PrivatekeyJWK`*) – The private key of the new entity
- **`parent_privatekey`** (*`PrivatekeyJWK`*) – The Private Key of caller (in this case the owner of "ala")

An example of the data in the request body:

```
{
    "DID": "did:elsi:VATES-B60645900",
    "domain_name": "in2.ala",
    "website": "www.in2.es",
    "commercial_name": "IN2 Innovating 2gether",
    "new_privatekey": {
        "kty": "EC",
        "crv": "secp256k1",
        "d": "Dqv3jmu8VNMKXWrHkppr5473sLMzWBczRhzdSdpxDfI",
        "x": "FTiW0a4r7S2SwjL7AlFlN1yJNWF--4_x3XTTxkFbJ9o",
        "y": "MmpxbQCOZ0L9U6rLLkD_U8LRGwYEHcoN-DPnEdlpt6A"
    },
    "parent_privatekey": {
        "kty": "EC",
        "crv": "secp256k1",
        "d": "Dqv3jmu8VNMKXWrHkppr5473sLMzWBczRhzdSdpxDfI",
        "x": "NKW_0Fs4iumEegzKoOH0Trwtje1sXsG9Z1949sA8Omo",
        "y": "g4B3EI0qIdlcXTn-2RpUxgVX-sxNFdqCQDD0aHztVkk"
    }
}
```

Response JSON Object

- **`didDocument`** (*`DIDDocument`*) – The DID document associated to the input DID

A more detailed explanation of each field follows:

`DID` is the DID of the new entity. We support ELSI DID method (ELSI_DID_Method) and AlastriaID. The DID has to be created before the call to the API with the appropriate method for the DID. In the case of ELSI this is trivial and described in the section mentioned above.

`domain_name` the domain name for the new entity in the Trust Framework. In the example it is *in2.ala* because it will be a sub-node of the Alastria one. The new identity will be created as a child node of the existing node owned by the entity controlling the `parent_privatekey`. If the parent domain name specified here is not owned by the entity controlling the `parent_privatekey`, an error is returned and no action is taken.

`website` the website address in the off-chain world, so other participants can look more information about the entity. This field is informational only. However, it can be used by external appications to check that the entity in th ereal world corresponds to the one registered in th eblockchain.

`commercial_name` the name of the company as it appears in the official register of the country/region. For example, in the case of IN2 (a Spanish business), the name should be the one registered in the Business Registry of Spain[3].

`new_privatekey` is the Private Key of the new entity, in JWK format. In this case the new entity is IN2. Please make sure the server being called is highly trusted.

`parent_privatekey` is the Private Key of the entity owning/controlling the parent node in the domain name, in JWK format. In this case the parent node is *ala*, corresponding to Alastria. Please make sure the server being called is highly trusted. Ideally, the server has to be operated by the same entity calling the API.

2.4 Credential flows

2.4.1 Credential Issuance

The figure below describes the interaction flows between the Issuer and the Citizen. Here the term Issuer includes the mobile application of the Issuer Operator and the associated backend system of the Issuer Entity.

The main interaction consists on the transmission of the Verifiable Credential from the Issuer to the mobile of the Citizen. The transmission is initiated with a QR.

The flows and the APIs used are described in detail below.

The credential issuance process is the following:

Credential generation

- The diagram assumes that the Issuer Operator starts the process for th ecreation of the credential, but other initiation mechanisms could be used depending on the context.

[3] http://www.rmc.es/Home.aspx?lang=en

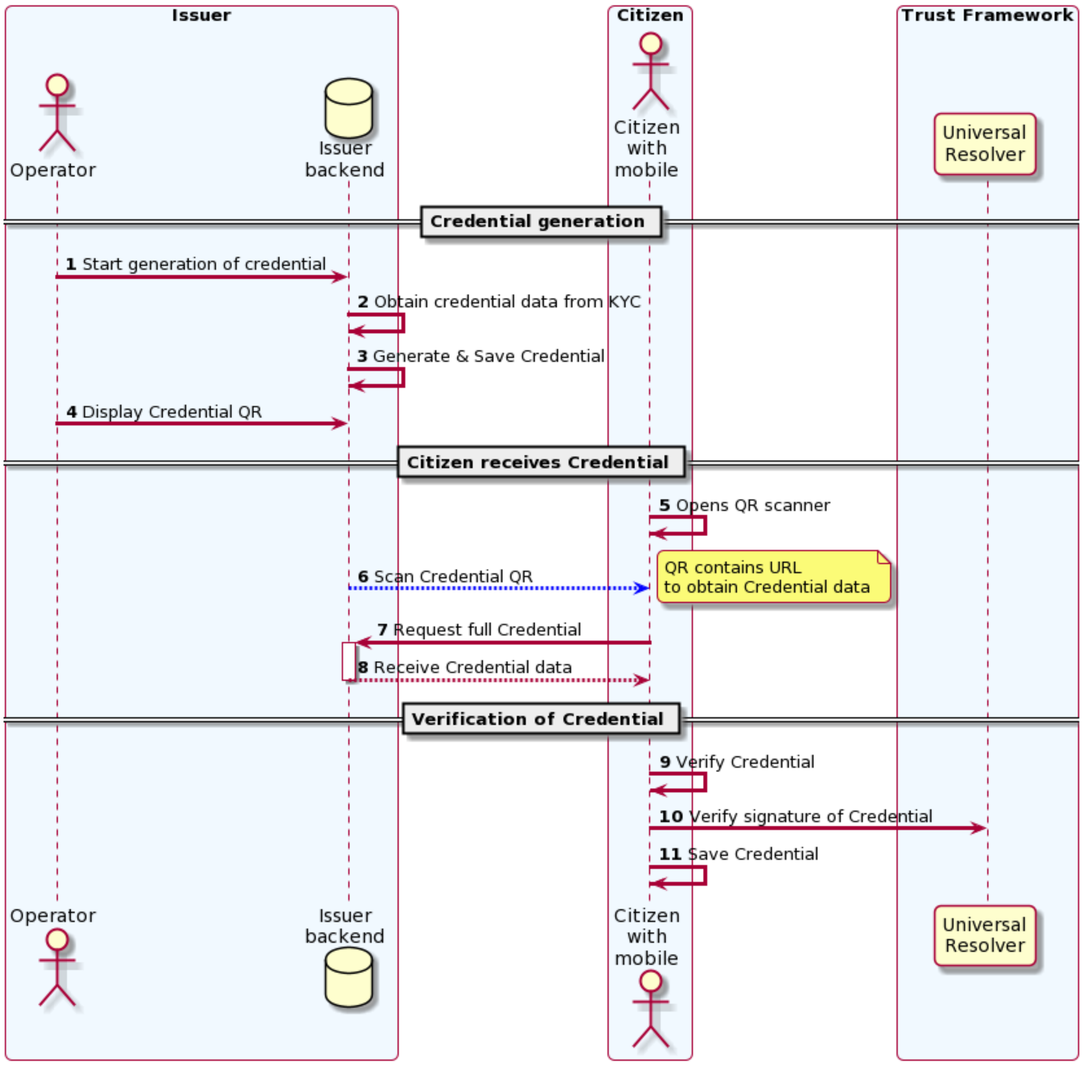

Issuer
Citizen
Trust Framework
Operator
Issuer backend
Citizen with mobile
Universal Resolver
Credential generation
1 Start generation of credential
2 Obtain credential data from KYC
3 Generate & Save Credential
4 Display Credential QR
Citizen receives Credential
5 Opens QR scanner
6 Scan Credential QR
QR contains URL to obtain Credential data
7 Request full Credential
8 Receive Credential data
Verification of Credential
9 Verify Credential
10 Verify signature of Credential
11 Save Credential
Operator
Issuer backend
Citizen with mobile
Universal Resolver

Fig. 5: Credential Issuance.

- The system gathers existing data from the citizen from a previous identification process, like KYC.
- The system stores the information and generates a credential in the standard W3C Verifiable Credential format.
- The system then generates and displays a QR code that will be scanned by the Citizen to receive the Credential. The QR contains the URL in the Issuer's system where the credential can be retrieved.

Citizen receives the Credential

- The Citizen uses the webapp to scan the QR code displayed by the Issuer Operator
- The Citizen mobile webapp uses the URL in the QR to get the credential in JWT format, signed by the Issuer.

Citizen webapp verifies the credential and signature of Issuer

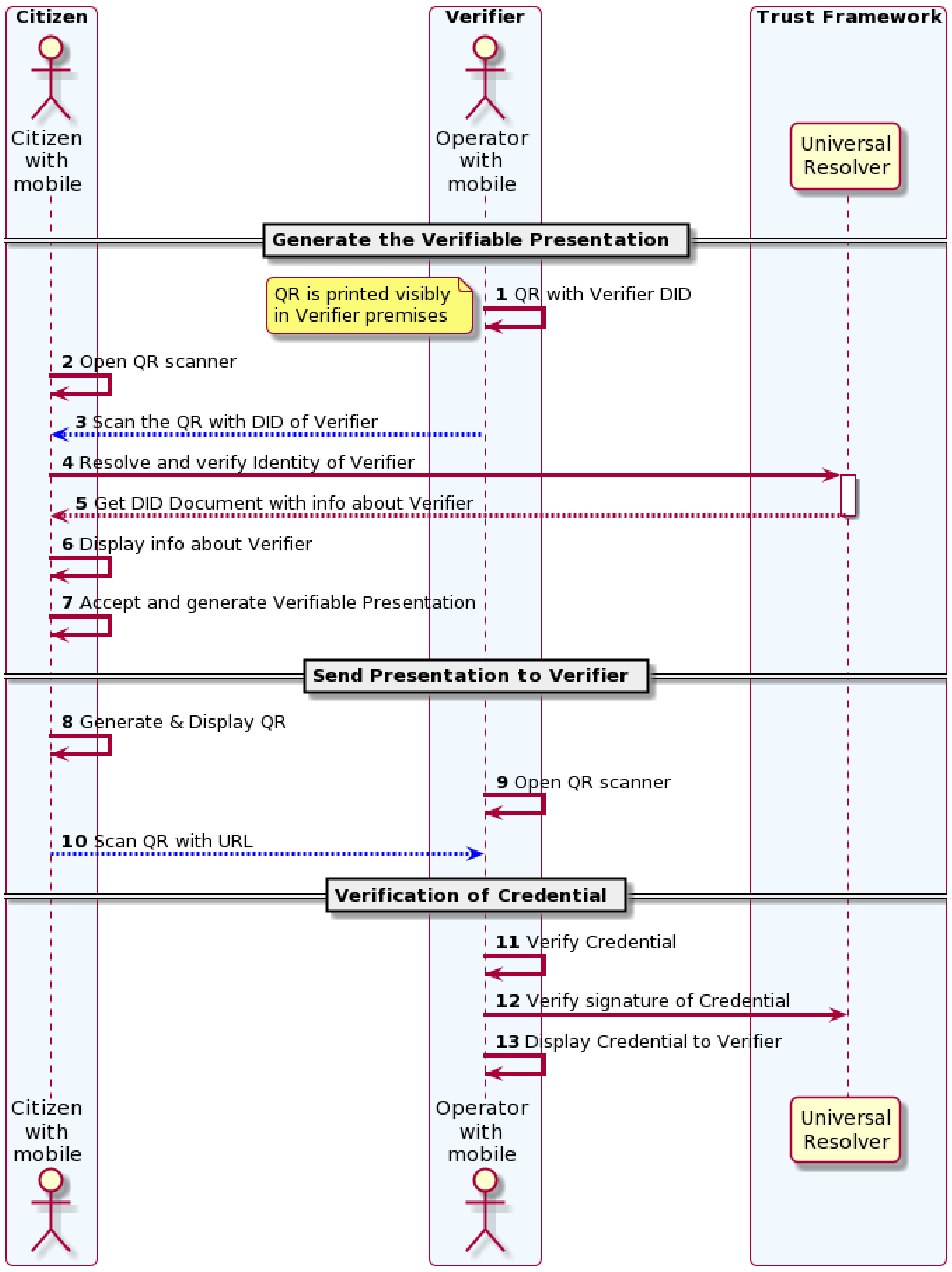
- The credential is verified as per the standard W3C Verifiable Credentials Implementation Guidelines[4].
- The verification includes resolving in the blockchain the identity of the Issuer Entity specified by the Issuer DID in the credential. The Issuer DID is registered in the blockchain and it includes the Public Key used by the Issuer Entity to digitally sign the credential.
- The Citizen mobile webapp uses a Universal Resolver to make this DID resolution and access the blockchain in read mode. The Universal Resolver is described in detail later in this document.
- After verification the credential is stored in the local storage of the Citizen mobile device. The user has also the option to store the credential in encrypted form in one or more of the personal cloud storage systems she has (Google Drive, MS Onedrive, Dorpbox, …).

2.4.2 Credential Verification

The system supports the standard online verification process as is common in most implementations of an SSI system. But in addition it supports a special flow for on-person verification of credentials, for example when the credential has to be presented to a Verifier Operator in-person and it has to be verified by the Operator. This flow is useful when some process has to be performed in-person in the offices of the Verifier Entity, or even when for some reason it has to be performed out of the offices. In other words, when the citizen is not interacting directly with a web page of the Verifier Entity.

This is the flow represented in the following diagram.

[4] https://w3c.github.io/vc-imp-guide



Citizen
Verifier
Trust Framework
Citizen with mobile
Operator with mobile
Universal Resolver
Generate the Verifiable Presentation
QR is printed visibly in Verifier premises
1 QR with Verifier DID
2 Open QR scanner
3 Scan the QR with DID of Verifier
4 Resolve and verify Identity of Verifier
5 Get DID Document with info about Verifier
6 Display info about Verifier
7 Accept and generate Verifiable Presentation
Send Presentation to Verifier
8 Generate & Display QR
9 Open QR scanner
10 Scan QR with URL
Verification of Credential
11 Verify Credential
12 Verify signature of Credential
13 Display Credential to Verifier
Citizen with mobile
Operator with mobile
Universal Resolver

Fig. 6: Credential Verification

2.5 ELSI: a DID Method for legal entities

The system supports several DID Methods using the Universal Resolver to resolve each DID into a corresponding DID Document. But the main DID Method used for legal persons, anchored into a Public-Permissioned blockchain, is *ELSI*: *did:elsi*.

2.5.1 ELSI DID syntax

The name ELSI stands for **E**TSI **L**egal person **S**emantics **I**dentifier, because it is based on the *Legal person semantic identifier* defined in the European Norm ETSI EN 319 412-1[5], related to digital signatures, peer entity authentication, data authentication as well as data confidentiality.

The ELSI DID Method refers only to legal persons, so we are using the *id-etsi-qcs-SemanticsId-Legal* definition described in Section 5.1 of ETSI EN 319 412-1.

Creating a DID is extremely simple and fully decentralized (does not require participation of any central authority), assuming that the legal person already exists. Its definition using ABNF syntax is:

```
did = "did:elsi:" id-etsi-qcs-SemanticsId-Legal
```

Which is the concatenation of the prefix *did:elsi:* with the legal person identifier defined in ETSI EN 319 412-1. For the full syntax, please refer to the standards document, but for the two most common basic identifiers (VAT and LEI) the identifier is composed of:

- 3 character legal person identity type reference, like *VAT* for identification based on a national value added tax identification number or *LEI* for the Legal Entity Identifier[6].
- 2 character ISO 3166 [2] country code;
- hyphen-minus "-" (0x2D (ASCII), U+002D (UTF-8)); and
- identifier (according to country and identity type reference).

Some examples of DIDs are the following:

Name	DID
ENDESA ENERGÍA (www.endesa.com)	**did:elsi:VATES-A81948077**
IN2 (www.ins.es)	**did:elsi:VATES-B60645900**
AENA (www.aena.es)	**did:elsi:VATES-A86212420**
Inter-American Development Bank (www.iadb.org)	**did:elsi:LEIXG-VKU1UKDS9E7LYLMACP54**
DAA plc (Dublin Airport Authority) (www.daa.ie)	**did:elsi:LEIXG-635400HRFGVKXFHZ8O77**

[5] https://www.etsi.org/deliver/etsi_en/319400_319499/31941201/01.04.02_20/en_31941201v010402a.pdf
[6] https://www.gleif.org

2.5.2 ELSI DID Document

An example DID Document is the following:

```
{
"payload": {
    "@context": [
    "https://www.w3.org/ns/did/v1",
    "https://w3id.org/security/v1"
    ],
    "id": "did:elsi:VATES-B60645900",
    "verificationMethod": [
    {
        "id": "did:elsi:VATES-B60645900#key-verification",
        "type": "JwsVerificationKey2020",
        "controller": "did:elsi:VATES-B60645900",
        "publicKeyJwk": {
        "kid": "key-verification",
        "kty": "EC",
        "crv": "secp256k1",
        "x": "3K4iNuzPkcrHlEbhHE8vYXlF6K5xGZ2rdOrn3cQ-LnQ",
        "y": "9Z_l_hQLkq6aLuZz8gheq7R_o5ZUHUlxZ3IBGHsdzaA"
        }
    }
    ],
    "service": [
    {
        "id": "did:elsi:VATES-B60645900#info",
        "type": "EntityCommercialInfo",
        "serviceEndpoint": "www.in2.es",
        "name": "IN2 Innovating 2gether"
    },
    {
        "id": "did:elsi:VATES-B60645900#sms",
        "type": "SecureMessagingService",
        "serviceEndpoint": "https://privatecred.hesusruiz.org/api"
    }
    ],
    "anchors": [
    {
        "id": "redt.alastria",
        "resolution": "UniversalResolver",
        "domain": "in2.ala",
        "ethereumAddress":
↪"0x8CDA8113567e633805e48c87747257E9FFAAdDF5"
    }
    ],
    "created": "2021-02-08T06:53:08Z",
    "updated": "2021-02-08T06:53:08Z"
}
}
```

2.6 PrivacyCred Verifiable Credentials

2.6.1 Data Model

The PrivacyCred credential uses the standard W3C Verifiable Credentials Data Model[7] for its representation, with some extensions to fit the requirements of this use case.

The specific credential data is encoded in the credentialSubject field of the VC. The following two figures represent the complete VC, where it has been divided in two parts to facilitate visualization.

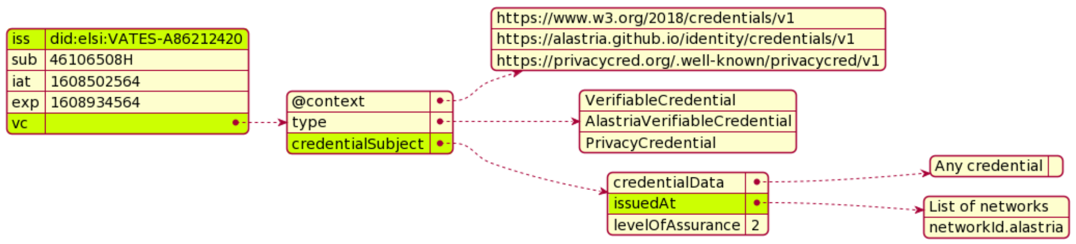


Fig. 7: W3C Verifiable Credential and extensions

The figure above represents the VC with standard fields and some extensions.

1. The iss field (issuer in VC terminology), uses the DID method `elsi`, specific for legal persons and explained in a section below.
2. There is an extension to specify the blockchain network (or networks) where the VC can be verified. More precisely, the `issuedAt` field of `credentialSubject` specifies the networks where the identity for the legal person that issued the credential can be verified.

   A legal person can have its *elsi* DID registered in one or more networks, and the same credential can be verified using any of those networks. The trust on the credential depends on the trust on the registration procedure of the identity of the signer. The Verifier entity can choose to verify the credential in whatever network is trusted to the Verifier.

   This mechanism provides a lot of flexibility in interoperability schemes across networks. More details are described in the section on interoperability.

2.6.2 Example of Verifiable Credential

```
{
    "exp": 1614770844,
    "iat": 1614252444,
    "iss": "did:elsi:VATES-X12345678X",
    "sub": "46106508H",
    "uuid": "829588b3162249d28f3eae5e84349777",
    "vc": {
            "@context": [
```

(continues on next page)

[7] https://www.w3.org/TR/vc-data-model

(continued from previous page)

```
                    "https://www.w3.org/2018/credentials/v1",
                    "https://alastria.github.io/identity/credentials/v1
↪",
                    "https://privacycred.org/.well-known/privacycred/v1"
                ],
                "type": [
                    "VerifiableCredential",
                    "AlastriaVerifiableCredential",
                    "PrivacyCredential"
                ],
                "credentialSchema": {
                    "id": "PrivacyCredential",
                    "type": "JsonSchemaValidator2018"
                },
                "credentialSubject": {
                    "privacyCredential": {
                            "citizen": {
                                "dob": "27-04-1982",
                                "idnumber": "46106508H",
                                "name": "COSTA/ALBERTO",
                                "type": "atRisk"
                            },
                            "comments": "These are some comments",
                    },
                    "issuedAt": [
                            "redt.alastria"
                    ],
                    "levelOfAssurance": 2
                }
        }
}
```

2.7 Verification of the credentials

The system includes two APIs to help client applications with the verification of credentials received from other actors in the ecosystem. The choice of API depends on the trust level of the client application on the server implementing the APIs

GET /api/did/v1/identifiers/(**string:** *DID*)

Resolves a DID and returns the DID Document (JSON format), if it exists. It supports four DID methods: **ebsi**, **elsi**, **ala**, **peer**.

Only **PEER** and **ELSI** (*https://github.com/hesusruiz/SafeIsland#62-elsi-a-novel-did-method-for-legal-entities*) are directly implemented by this API. The others are delegated to be resolved by their respective implementations.

For example, for **EBSI** we call the corresponding Universal Resolver API, currently in testing and available at https://api.ebsi.xyz/did/v1/identifiers/{did}

Query Parameters

- **DID** (*string*) – The DID to resolve into a DID Document.

Response JSON Object

- **didDocument** (*payload*) – The DID document associated to the input DID

Status Codes

- 200 OK[8] – no error
- 404 Not Found[9] – error resolving the DID

Example request:

```
GET /api/did/v1/identifiers/did:elsi:VATES-B60645900 HTTP/1.1
Host: example.com
Accept: application/json
```

Example response:

```
HTTP/1.1 200 OK
Vary: Accept
Content-Type: text/javascript

{
    "payload": {
        "@context": [
            "https://www.w3.org/ns/did/v1",
            "https://w3id.org/security/v1"
        ],
        "id": "did:elsi:VATES-B60645900",
        "verificationMethod": [
            {
                "id": "did:elsi:VATES-B60645900#key-verification
↪",
                "type": "JwsVerificationKey2020",
                "controller": "did:elsi:VATES-B60645900",
                "publicKeyJwk": {
                    "kid": "key-verification",
                    "kty": "EC",
                    "crv": "secp256k1",
                    "x":
↪"3K4iNuzPkcrHlEbhHE8vYXlF6K5xGZ2rdOrn3cQ-LnQ",
                    "y": "9Z_l_hQLkq6aLuZz8gheq7R_
↪o5ZUHUlxZ3IBGHsdzaA"
                }
            }
        ],
        "service": [
            {
```

(continues on next page)

(continued from previous page)

```
                    "id": "did:elsi:VATES-B60645900#info",
                    "type": "EntityCommercialInfo",
                    "serviceEndpoint": "www.in2.es",
                    "name": "IN2 Innovating 2gether"
                },
                {
                    "id": "did:elsi:VATES-B60645900#sms",
                    "type": "SecureMessagingService",
                    "serviceEndpoint": "https://privatecred.
↪hesusruiz.org/api"
                }
            ],
            "anchors": [
                {
                    "id": "redt.alastria",
                    "resolution": "UniversalResolver",
                    "domain": "in2.ala",
                    "ethereumAddress":
↪"0x8CDA8113567e633805e48c87747257E9FFAAdDF5"
                }
            ],
            "created": "2021-02-08T06:53:08Z",
            "updated": "2021-02-08T06:53:08Z"
        }
}
```

In general, validating a credential involves the following:

1. Deserialize the JWT without verifying it (we do not yet have the public key).
2. Get the `kid` property from the header (the JOSE header of the JWT).
3. The `kid` has the format did#id where `did` is the DID of the issuer and `id` is the identifier of the key in the DIDDocument associated to the DID.
4. Perform resolution of the DID of the issuer with the Universal Resolver API.
5. Get the public key specified inside the DIDDocument.
6. Verify the JWT using the public key associated to the DID.
7. Verify that the DID in the `iss` field of the JWT payload is the same as the one that signed the JWT.

`POST /api/verifiable-credential/v1/verifiable-credential-validations`

Is the easiest one to use and the one requiring higher level of trust. The client app just passes the JWT in the JWS Compact Serialization format (RFC 7519) as the body of a POST request and the server verifies the credential and credential signature using internally the Universal Resolver API for resolving the DID of the Issuer and checking its digital signature.

[8] http://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html#sec10.2.1
[9] http://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html#sec10.4.5

Request JSON Object

- **`credential`** (*`JWT`*) – The credential in JWT format.

Response JSON Object

- **`claims`** (*`object`*) – The JSON object with the verified claims in the JWT. Otherwise, an error

Status Codes

- 200 OK[10] – no error
- 404 Not Found[11] – error resolving the DID

The easiest one to use is `/api/verifiable-credential/v1/verifiable-credential-validations`, and it is the one requiring higher level of trust. The client app just passes the JWT in the JWS Compact Serialization format (RFC 7519) as the body of a POST request and the server verifies the credential and credential signature using internally the Universal Resolver API for resolving the DID of the Issuer and checking its digital signature.

`/api/did/v1/identifiers/(string:DID)` is the Universal Resolver API. The client application will have to perform the validations that the server does in the previous call.

[10] http://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html#sec10.2.1
[11] http://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html#sec10.4.5