**INTRODUCTION TO DID**

**Decentralized Identifiers :**

Decentralized Identifiers (DIDs) are a new type of identifier that enable verifiable, self-sovereign digital identities. Unlike traditional identifiers such as email addresses or usernames, DIDs are created, owned, and controlled by the individual. This means there is no need to rely on a central authority like a government or a corporation to manage these identifiers.  
  
**Key Features of DIDs**

1. **Decentralization**: DIDs are designed to be used across different platforms and systems without requiring a central registry. This reduces dependency on any single entity and enhances security and privacy.
2. **Self-Sovereign**: Individuals have full control over their DIDs. They can create, manage, and use their own digital identities without needing permission from any external party.
3. **Interoperability**: DIDs can be used across various services and platforms, making them highly flexible and adaptable for different use cases.
4. **Security**: DIDs often employ blockchain technology and cryptographic proofs to ensure the security and integrity of the identity information.

**How Do DIDs Work?**

* **Creation**: Individuals or entities can create a DID using a decentralized system like a blockchain. Each DID is unique and can be associated with a public-private key pair.
* **Ownership**: The owner of a DID controls it using their private key. They can use this private key to sign digital documents or transactions, proving ownership and authenticity.
* **Verification**: Other parties can verify the validity of a DID by checking the associated public key and any cryptographic proofs. This ensures that the identity information is trustworthy and has not been tampered with.

**Introduction to Privado ID**

Privado ID is a cutting-edge identity infrastructure that facilitates trusted and secure relationships between apps and users, adhering to the principles of self-sovereign identity and privacy by default. It provides a robust platform for organizations to issue and verify verifiable credentials. Privado ID enables organizations on one side to issue verifiable credentials about users, and organizations, on the other side, to verify those claims via a suite of tools created for each member of the SSI ecosystem.

**Key Features of Privado ID**

1. **Zero-Knowledge Proofs**: Privado ID uses advanced cryptographic techniques, allowing users to prove their identity without exposing private information.
2. **Privacy by Default**: Users' identities are secured by zero-knowledge cryptography, ensuring that personal information remains private.
3. **Trusted Relationships**: Privado ID enables organizations to issue and verify credentials, fostering secure and trustworthy interactions.

**How Privado ID Works**

**Verifiable Credentials, Identity Holder, Issuer, and Verifier**

Every identity is associated with a unique identifier known as a DID (Decentralized Identifier). All identity-related information is represented through Verifiable Credentials (VCs). Simply put, a VC is any type of information about an individual, organization, or object. This information can range from basic details like age or educational qualifications to specific credentials like a membership certificate issued by a DAO.

The framework's architecture consists of three main components: Identity Holder, Issuer, and Verifier. Together, these components form what is known as the Triangle of Trust. Let's explore the roles each entity plays in Privado ID.

**Identity Holder**

An Identity Holder is an entity that keeps claims in its digital wallet. A Verifiable Credential (VC) is issued to the Identity Holder by an Issuer. The Identity Holder generates zero-knowledge proofs of the VCs and presents these proofs to the Verifier. The Verifier then checks the authenticity of the proof and ensures it matches specific criteria.

**Issuer**

An Issuer is an entity (which can be a person, organization, or object) that issues VCs to Identity Holders. Each VC is cryptographically signed by the Issuer to ensure its authenticity. Every VC originates from an Issuer.

**Verifier**

A Verifier is responsible for verifying the proofs presented by the Identity Holder. It requests the Identity Holder to provide a proof based on the VCs stored in their wallet. During the verification process, the Verifier performs various checks, such as confirming that the VC was signed by the expected Issuer and ensuring that the VC meets the requested criteria. This verification can occur either off-chain or on-chain

The simplest example of a Verifier is a bar that wants to verify if a client is over 18. In the real world, the Identity Holder would need to provide an ID and show all their personal information. With Privado ID, they only need to pass a proof.  
  
A diagram of a company's identity

Description automatically generated

**Role of a Wallet**

A Wallet is essential for the smooth exchange of Verifiable Credentials (VCs) with the Issuer and the presentation of proofs to the Verifier. An Identity Holder keeps their personal data, represented as VCs, within their wallet. The primary functions of the wallet include storing the user's private key, retrieving VCs from the Issuer, and generating zero-knowledge proofs to be presented to the Verifier.  
  
  
**Quick Start**

**How Privado ID Works: A Simple Example**

To show you how Privado ID works, we'll use a simple example involving a tool called POAP, which stands for Proof of Attendance Protocol. POAP is used to prove that someone attended a specific event.

In this guide, we’ll briefly explain the three key roles in the Privado ID system: the Identity Holder, the Issuer, and the Verifier. We'll use the example of a person who needs to prove that they attended a particular event.  
  
  
**Steps :**

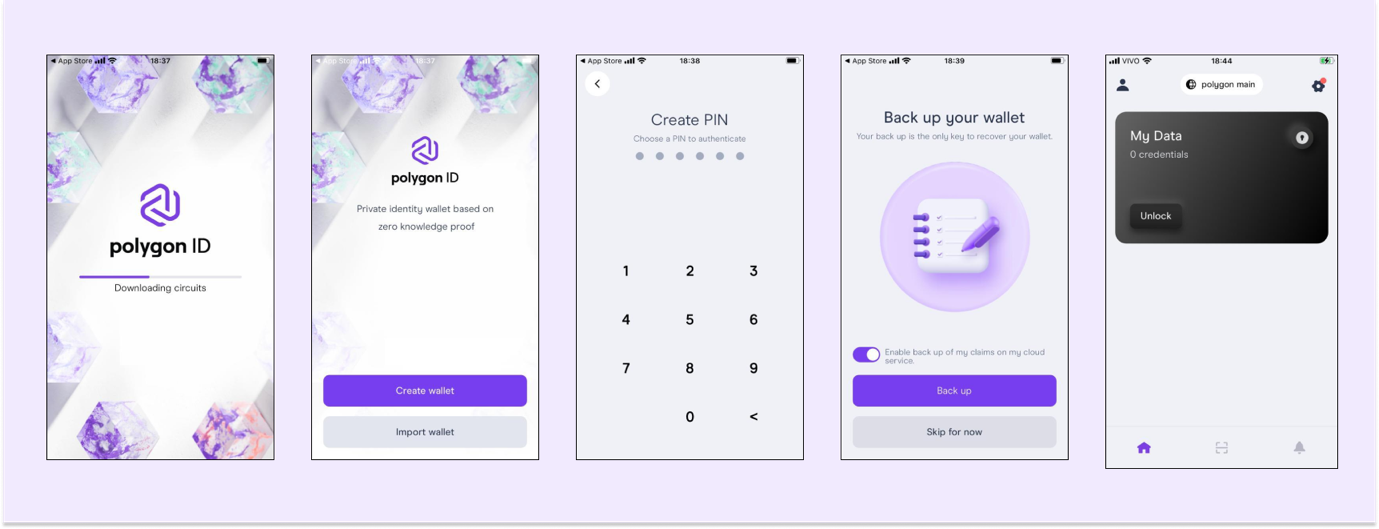
1.Set up a Polygon ID wallet.

2.Issue a new credential to attest to the ID Holder's attendance at the event.

3.Fetch the newly created credential.

4.Verify the credential's validity.

**Set up a Polygon ID wallet**

As an identity holder, individual who wants to have a credential to prove his age, for example, will need an application that can hold their credentials. In our case, we will be using the Polygon ID Wallet.  
  
**To get started with the Polygon ID Wallet, download the Polygon ID Wallet App and create an Identity**  
The process from downloading to creating an identity on the Polygon ID Wallet is just as it is shown below. You need to download the app, create a wallet, set up a PIN number and the wallet is ready to be used.  
  


**Issue the credential**[**​**](https://docs.privado.id/docs/quick-start-demo#issue-the-credential)

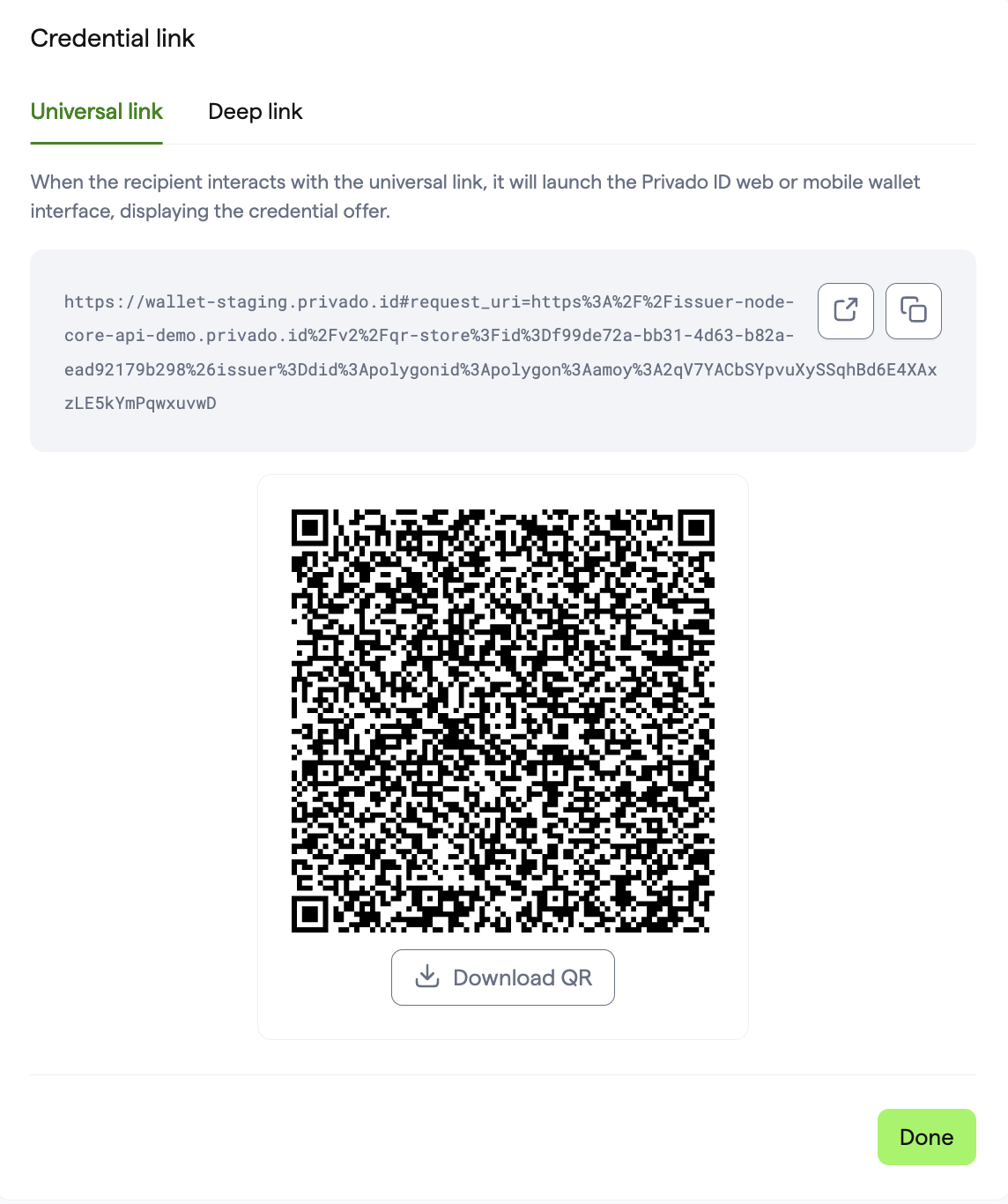
**With the new schema in hand, the issuer should now be able to generate a credential.**

1. First, go to the Issuer Node UI testing environment.
2. Now you need to import the schema. Click on Import Schema and paste our previously generated schema IPFS address ipfs://QmTSwnuCB9grYMB2z5EKXDagfChurK5MiMCS6efrRbsyVX:A screenshot of a computer

   Description automatically generated

1. You can go ahead and click on Issue Credential in the top righ-hand corner. Choose Credential Link on the next page and your schema on the dropdown menu ("POAP01", in our case). For this credential, we are providing a proof of attendance to an event in Paris:  
     
   A screenshot of a computer

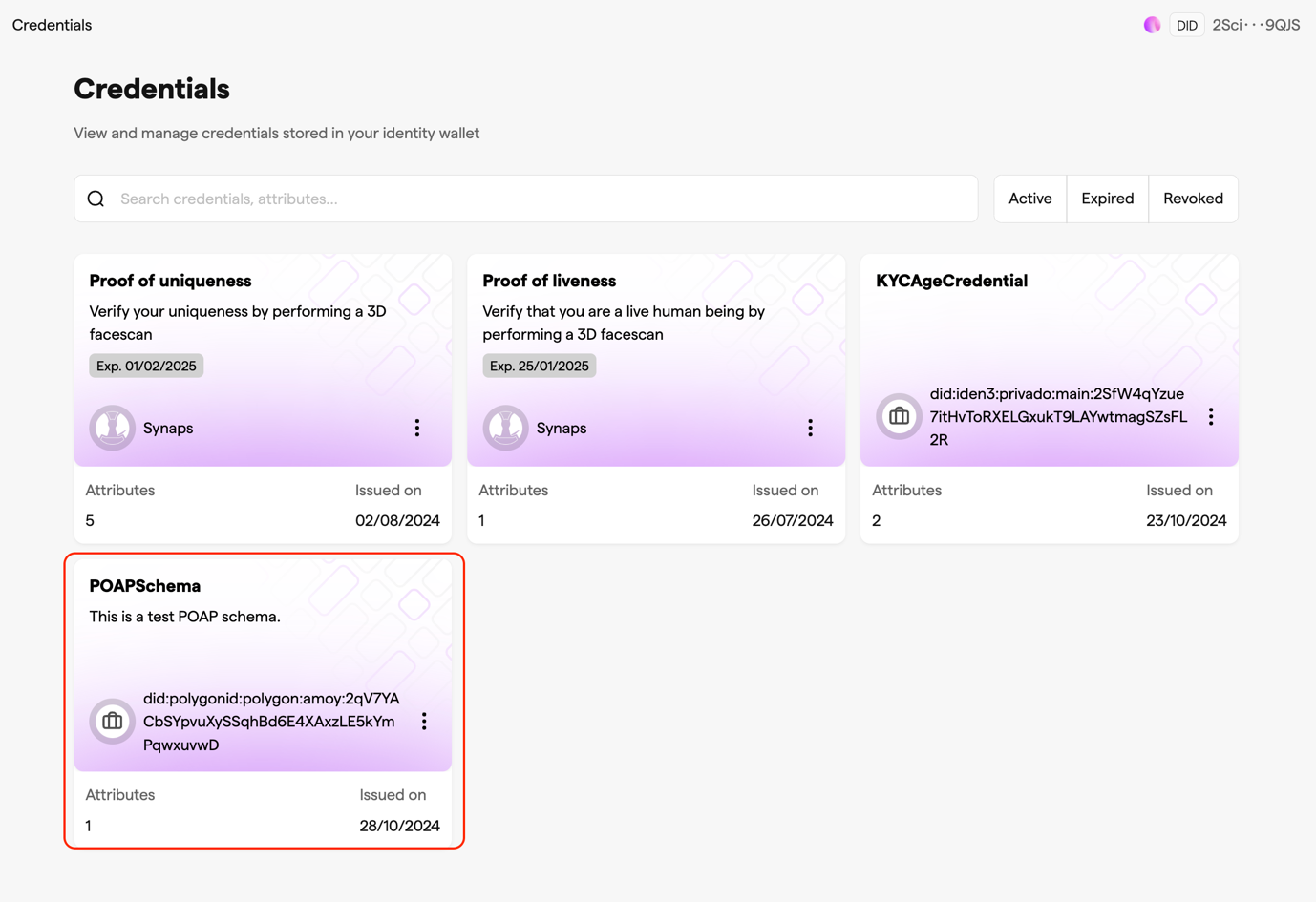
   Description automatically generated
2. After you click on create credential link, you can also click on View link on the next screen to check the generated QR Code



**Fetch the newly created credential**

Obtain the newly created credential. We are now returning to the role of the ID Holder. You can use either the Web Wallet in your browser or the Wallet App to accept the credential through the link or QR code provided by the issuer in the previous step.  
  
**Web Wallet**

After clicking the Universal link, you will be directed to the Web Wallet. Click "Sign In" and connect your crypto wallet, such as Metamask. Then, click "Add to my wallet" to add the credential to your wallet.  
  
Screens screenshot of a screenshot of a computer

Description automatically generated  
  
Click on **Manage your credentials** to view and manage the credential in the Web Wallet.  
  


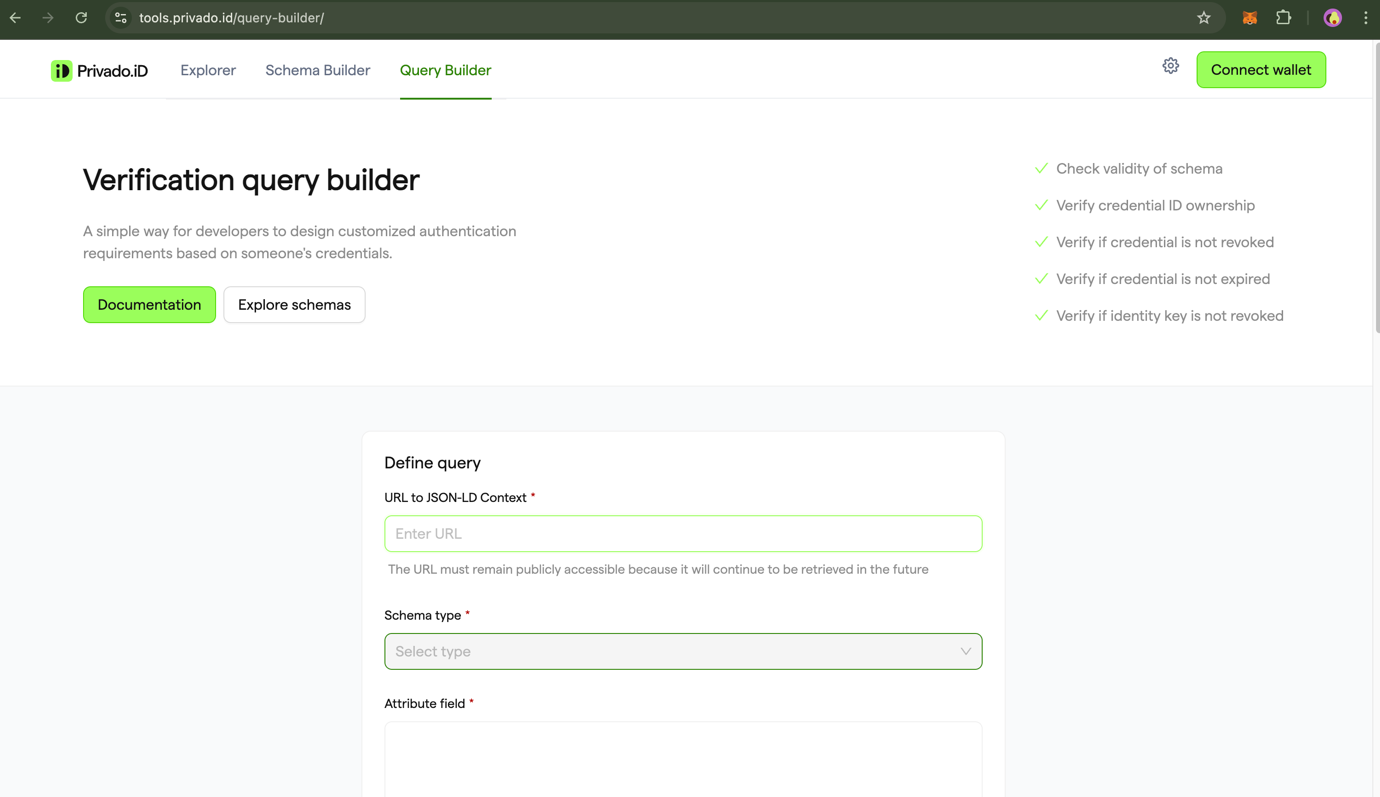
**Mobile Wallet App**[**​**](https://docs.privado.id/docs/quick-start-demo#mobile-wallet-app)

You can scan the QR code using the Wallet App. Then, click Sign In. This will authenticate and add the credential to your Wallet.  
  
Screens screenshot of a phone

Description automatically generated

**Verify the ID holder credential**

Introducing the third role in this tutorial: the verifier. This role is typically represented by an organization that needs to confirm certain details of an individual's credentials. In our scenario, this organization aims to verify whether the ID holder attended our fictional Paris event.

Follow these steps to verify the credential:  
  
1. Navigate to the Query Builder website. This tool is specifically designed to streamline the creation of verification queries.  
  
  
  
2. Now, you need to define the query. Utilize the provided JSON-LD URL: ipfs://QmdH1Vu79p2NcZLFbHxzJnLuUHJiMZnBeT7SNpLaqK7k9X. The query should be structured as follows:  
  
A screenshot of a phone

Description automatically generated

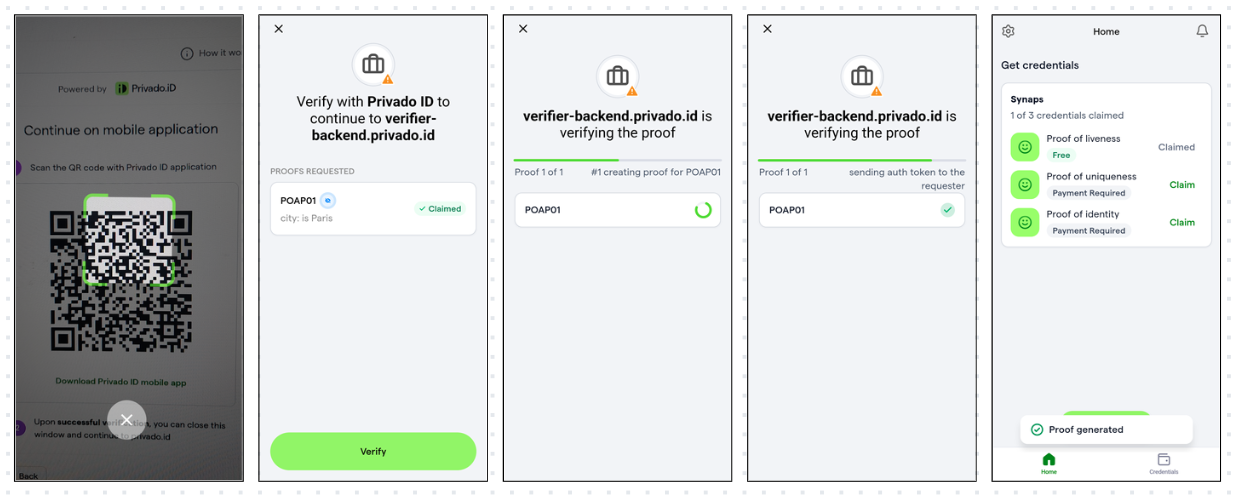
3. Click on "Create Query." Then, click the "Test Query" button, which will redirect you to the Privado ID Web Wallet.  
  
A screenshot of a computer

Description automatically generated  
  
4. Click "Sign In" and connect your crypto wallet. Since you have already claimed the credential, it will be displayed as 'claimed'. Click "Verify" to initiate the proof generation process  
  
A screenshot of a screenshot of a computer

Description automatically generated  
  
5. Finally, the proof is generated and sent to the verifier. The verifier will then check the revocation status and any additional information to validate the proof. You will see the following response on the Query Builder website:

A screenshot of a computer

Description automatically generated  
  
Alternatively, you can use the Privado ID Wallet app for verification. After Step 3, when you are redirected to the Web Wallet, click on "Continue via app" to display a QR code. Open the Privado ID Wallet App, scan the QR code, and then click "Verify."



The above guide illustrates the basic functionalities of Privado ID using a POAP use case, which includes wallet setup, credential issuance, retrieval, and verification. Although this is a simplified example, Privado ID offers a complete suite of SSI tools for managing decentralized identity and verifiable credentials.

**Issuer**

An Issuer is any entity that provides Verifiable Credentials. You can think of a credential as a declaration: something an Issuer asserts about another entity. For example, when a university (Issuer) declares that a student (subject) has earned a degree, this is a credential.

An Issuer can be:

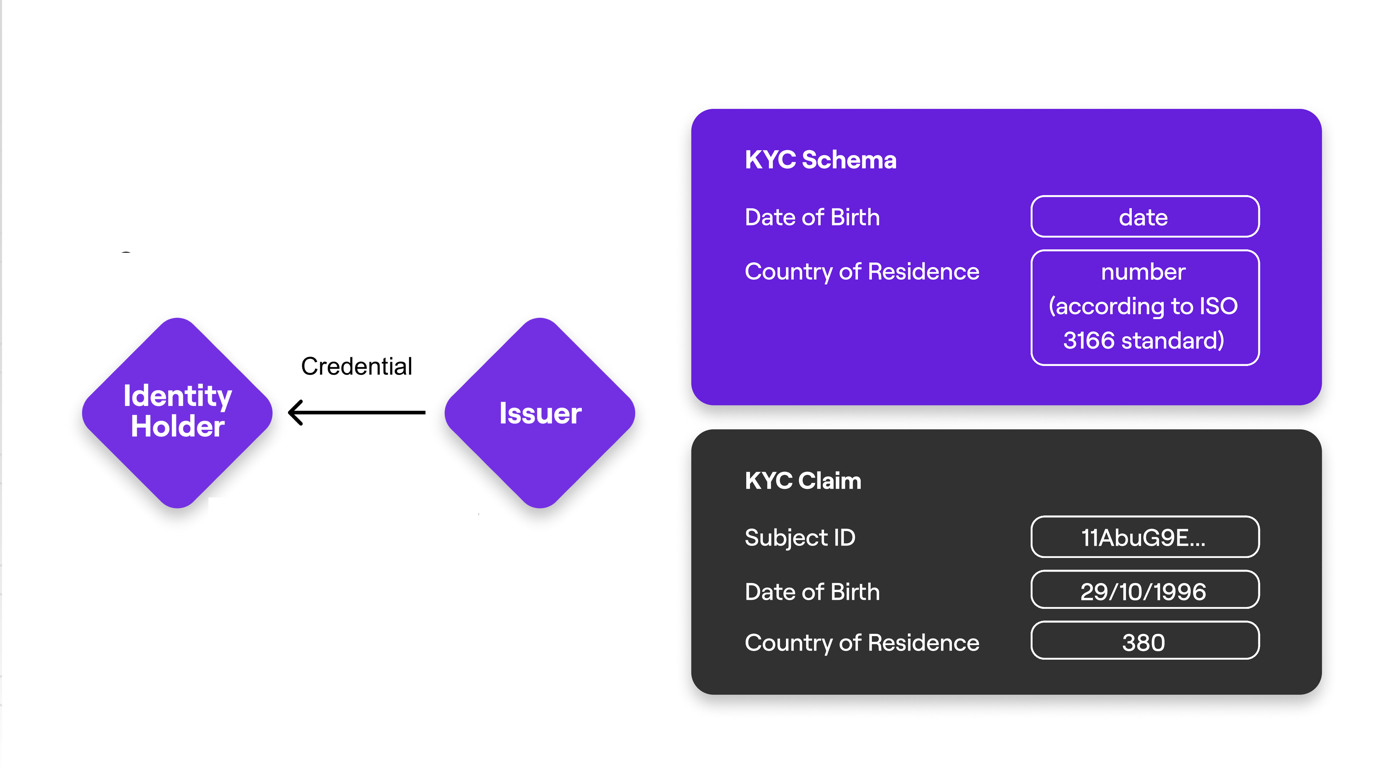
* A DAO that grants "membership claims" to its members.
* A government institution that issues identity documents to its citizens.
* A facial recognition machine learning application that provides "proof of personhood" claims.
* An employer that endorses its employees.

**Operating an Issuer**

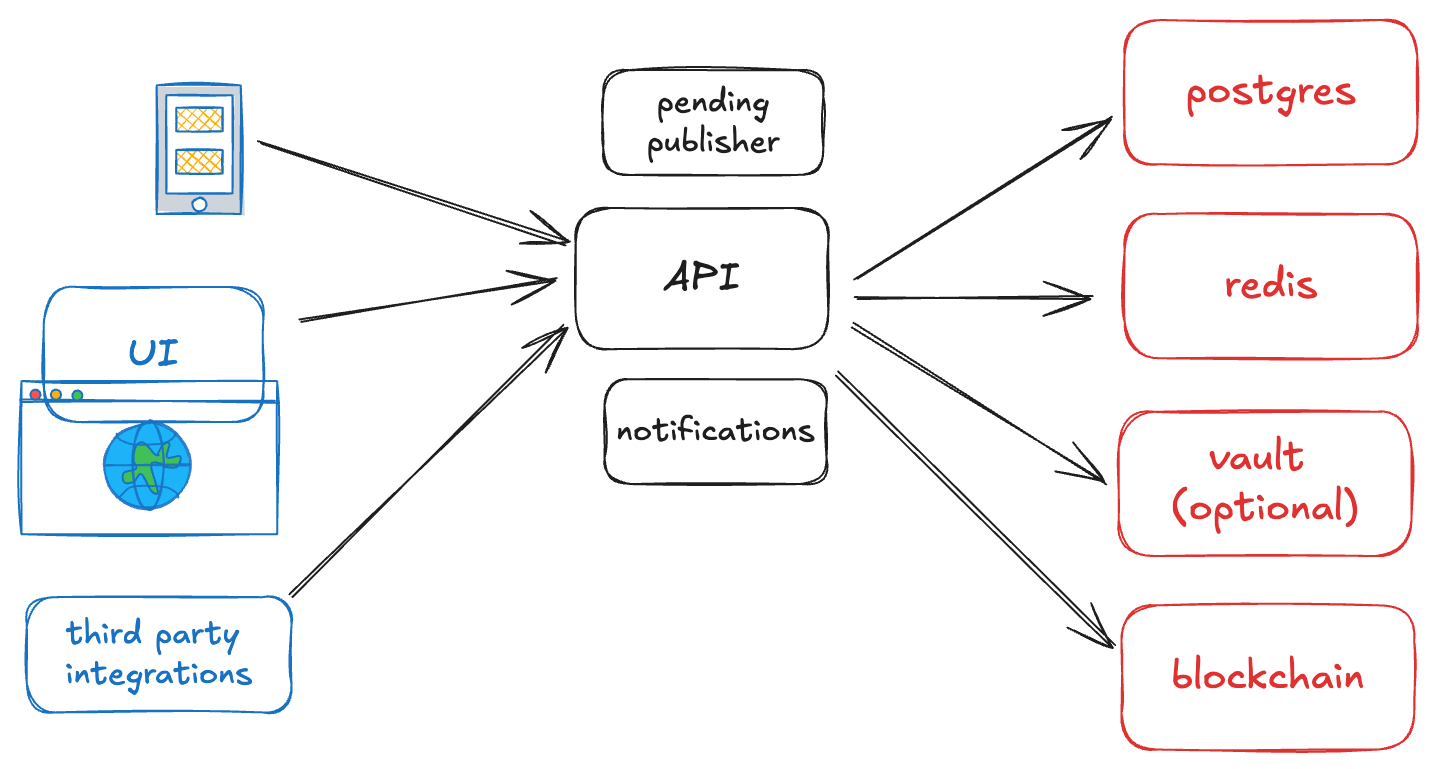
There are several methods to handle issuer-related tasks, such as managing and issuing credentials and establishing connections with holders. Here are the currently available options:

1. Running an Issuer Node directly within your own infrastructure.
2. Integrating the JS SDK into your application to issue credentials.
3. Utilizing SaaS vendors that use Privado ID solutions.
4. Employing Privado ID smart contracts for on-chain credential issuance.

**Issuer Nodes**

To function, an Issuer needs to operate an Issuer Node, which is a self-hosted node that provides all the essential functionalities for performing issuer tasks.  
  
  
There are basically two ways the Issuer Node can be operated:

* Issuer Node API
* Issuer Node UI



**Issuer Node API**[**​**](https://docs.privado.id/docs/issuer/issuer-overview#issuer-node-api)

The Issuer Node API is tailored for developers and integrators who want to create custom solutions using Privado ID's features. It is perfect for those who need access to detailed, low-level information, such as Merkle Trees.  
  
For more information on Issuer node API, please refer to :  
https://docs.privado.id/docs/category/issuer-node-api/  
  
**Issuer Node UI**

The Issuer Node UI offers a comprehensive experience with an easy-to-use interface for managing an Issuer Node's full capabilities. It simplifies essential tasks such as issuing and revoking credentials, publishing states, and managing schemas, connections, and identities. Powered by the Issuer Node API, it provides full control over these features without the need for direct API interaction.  
  
For more information on Issuer Node UI, please refer to :  
https://docs.privado.id/docs/category/issuer-node-ui/  
  
**Features**

**Revocation Status**[**​**](https://docs.privado.id/docs/issuer/features#revocation-status)

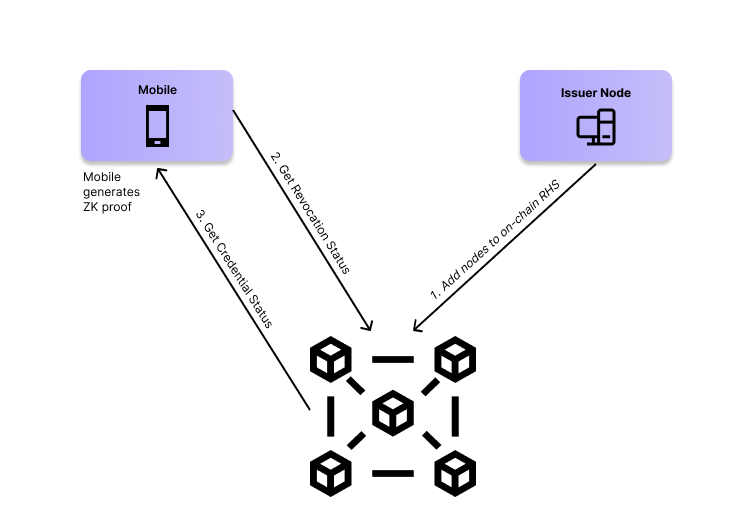
Mobile applications can obtain the revocation status of a specific credential by directly requesting this information from the Issuer Node. However, if the Issuer Node is offline, this communication will not function. To address this issue, the Reverse Hash Service and the On-chain Revocation Status provide two viable solutions.  
  
A diagram of a process

Description automatically generated  
  
**Reverse Hash Service**[**​**](https://docs.privado.id/docs/issuer/features#reverse-hash-service)

The Reverse Hash Service (RHS) stores all the revocation information online which can be accessed by mobile applications and verifiers. In this scenario, the communication between mobile and Reverse Hash Service replaces the link between the application and the Issuer Node:  
  
A diagram of a diagram

Description automatically generated  
  
As the diagram shows, the Issuer Node sends the revocation information to the RHS. The service, then, is able to return the credential status to the application.

**On-chain Revocation**

The On-chain Revocation Status utilizes blockchain decentralized storage to permanently store revocation information on the network. This ensures that, even if the Issuer Node or the Reverse Hash Service is unavailable, the data remains accessible on-chain.  
  
  
  
**Verifier**

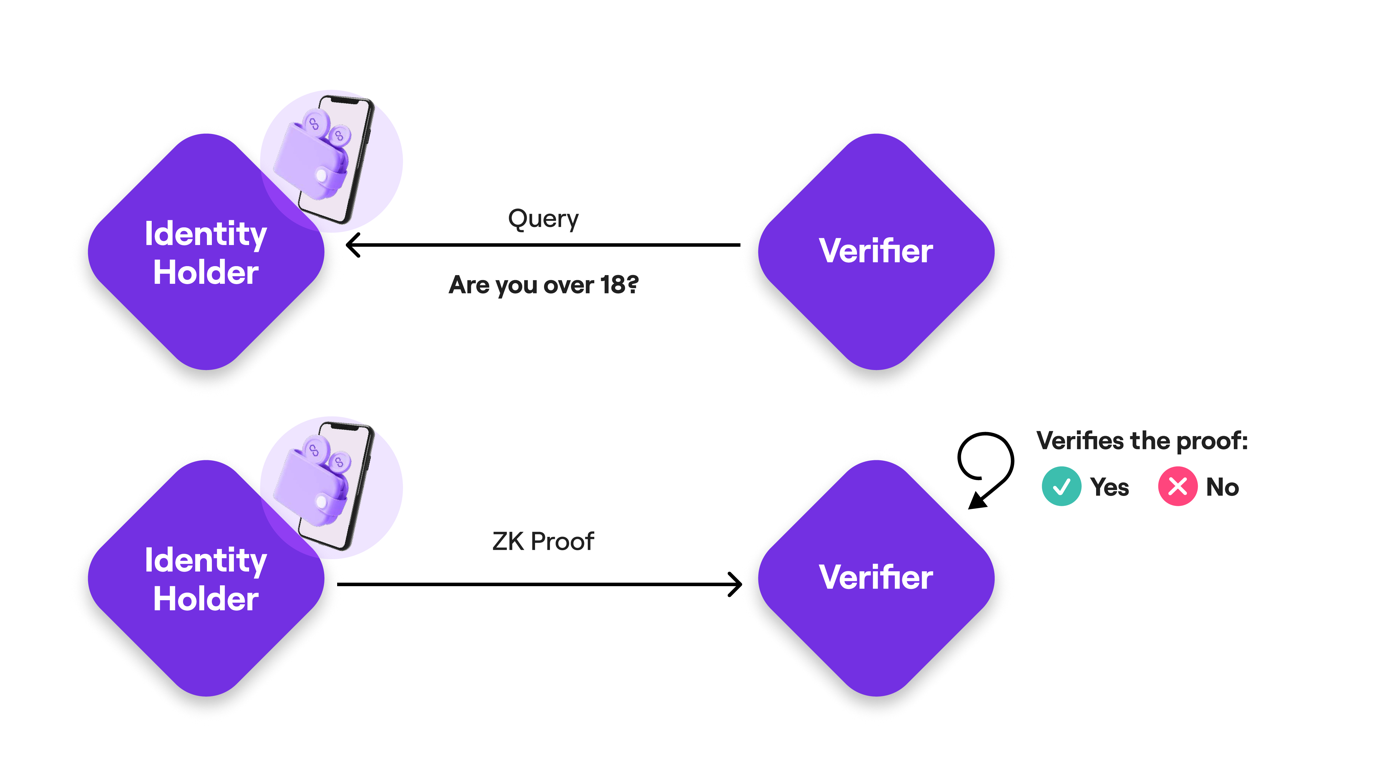
A Verifier is any web2 or web3 platform that wants to authenticate users based on their Credentials.

Verifiers can create queries using users’ existing Credentials collected from various Issuers. A Query specifies the criteria that a user must meet to authenticate, such as "must be a member of XYZ DAO" or "must be over 18 years old." Privado ID offers a seamless, customized, and privacy-focused authentication experience for users.

The Verifier's request is crafted using our fully expressive zk Query Language and is encapsulated into a Universal Link or QR code for the user to access. The user clicks the Universal Link or scans the QR code with their mobile wallet app to generate the proof.

The verification process does not require any interaction between the Verifier and the Issuer of the requested credential. As part of the Query, the Verifier includes the identifiers of the trusted issuers. For instance, a Verifier should include XYZ DAO as the only trusted Issuer when verifying that an individual is a member of XYZ DAO. The XYZ DAO does not need to acknowledge or interact with the Verifier.

At the end of the process, the Verifier receives a cryptographic proof that the user meets the query criteria, while the user shares only the minimum amount of data necessary for the interaction



**Verifier SDK**

The Verifier SDK is a collection of tools that enable any application to verify user information based on their credentials. It offers a fully customizable and privacy-focused authentication experience for users. Verification of user information can occur either on-chain via a smart contract or off-chain, both ensuring the same level of user privacy and query customization. The proof generated on the user's mobile device is identical in both scenarios; the only difference lies in the verification process. On-chain verification is conducted programmatically within a smart contract, while off-chain verification is executed within a script set up by the Verifier application, either on a server or client side.

* **Verifier SDK - Off-chain Verification**: Provides all the necessary components to create a customized query, set up a verifier, and generate a Universal Link or QR code (or deep link) on the client side to request proof from the user.  
  For more information on Off-Chain Verification, please refer to : https://docs.privado.id/docs/category/off-chain-verification/
* **Verifier SDK - On-chain Verification**: Allows decentralized applications (Dapps) to verify users' credentials within a Smart Contract using zero-knowledge proof cryptography.  
  For more information on On-Chain Verification, please refer to : https://docs.privado.id/docs/category/on-chain-verification

**Features**

**Selective Disclosure**

There are two types of proof requests: private proof and selective disclosure.

* **Private Proof Requests**: In these requests, the credential attribute values are not disclosed. The response is simply a true/false answer to the requested value.
* **Selective Disclosure Requests**: In these requests, the verifier asks for a specific attribute value from the credential to be disclosed.

The process for selective disclosure is almost identical to that of a private proof request. The only difference is that the query in the QR code from the verifier's proof request has a different format. The SDK handles this by calling the authenticate method from iden3comm.dart with the iden3MessageEntity as a parameter.