EBSI-VECTOR
Education and work reloaded

D3.6: Open-source reference implementation of an issuer and verifier wallet for EBSI. Earlier release of the quick wins in a parallel track.

Project title:	**EBSI-VECTOR** - Open-source reference implementation of an issuer and verifier wallet for EBSI. Earlier release of the quick wins in a parallel track.
Grant Agreement No.	101102512 - DIGITAL-2022-DEPLOY-02-EBSI-SERVICES
Deliverable Title	D3.6 – Open-source reference implementation of an issuer and verifier wallet for EBSI. Earlier release of the quick wins in a parallel track.
Version:	1.6
Date:	08/11/2024
Responsible Partner:	IZERTIS
Authors:	Florentín Pérez González (IZERTIS)
Contributing Partners:	IGRN, PROTOKOL, VALIDATE ID, walt.id, INAIL, Goldman, NASK, CERTSIGN, SKS, GUNET, FNMT, DANUBETECH, CERTH, Gataca, FNMT, ENG
Reviewers:	Kristina Livitckaia (CERTH), Ioannis Grypiotis (CERTH), Andrei Ternauciuc (UPT), Carmen Holotescu (UPT), Fernando Mata Martín (FNMT)
Dissemination Level:	PU – Public

Project co-funded by the European Union under the Digital Europe Programme. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union. Neither the European Union nor the granting authority can be held responsible for them.

Document Change History

Version	Date	Author (organisation)	Description
V0.1	10/10/2024	Florentín Pérez González (Izertis)	Architecture and capabilities.
V0.2	11/10/2024	Florentín Pérez González (Izertis)	Data Model
V1.0	14/10/2024	Florentín Pérez González (Izertis)	First version
V1.1	16/10/2024	Florentín Pérez González (Izertis)	Expanded "Same Device and Cross Device Support" section with VP Verifications QRs/deeplink
V1.2	28/10/2024	Florentín Pérez González (Izertis)	License and added next steps in the conclusion
V1.3	29/10/2024	Kristina Livitckaia & Ioannis Grypiotis (CERTH)	Internal review
V1.4	31/10/2024	Andrei Ternauciuc (UPT) & Carmen Holotescu (UPT)	Internal review
V1.5	31/10/2024	Fernando Mata Martín (FNMT)	Internal review
V1.6	08/11/2024	Florentín Pérez González (Izertis), Vincenzo Savarino (ENG)	Final Version

Project co-funded by the European Union under the Digital Europe Programme. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union. Neither the European Union nor the granting authority can be held responsible for them.

Table of Contents

Project co-funded by the European Union under the Digital Europe Programme. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union. Neither the European Union nor the granting authority can be held responsible for them.

Project co-funded by the European Union under the Digital Europe Programme. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union. Neither the European Union nor the granting authority can be held responsible for them.

List of Figures

List of Tables

Project co-funded by the European Union under the Digital Europe Programme. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union. Neither the European Union nor the granting authority can be held responsible for them.

List of Terms and Abbreviations

Abbreviation	Definition
VC	Verifiable Credential
VP	Verifiable Presentation
OID4VCI	OpenidID for Verifiable Credential Issuance
OID4VP	OpenidID for Verifiable Presentation
EBSI	European Blockchain Infrastructure
API	Application Programming Interface
DID	Decentralized Identifier
DIF	Decentralized Identity Foundation
W3C	World Wide Web Consortium
RFC	Request For Comments
AGPL	Affero General Public License
QR Code	Quick Response Code
JWK	JSON Web Key
URL	Uniform Resource Locator
WP	Work Package
REST API	Representational State Transfer API
GUI	Graphical User Interface

Project co-funded by the European Union under the Digital Europe Programme. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union. Neither the European Union nor the granting authority can be held responsible for them.

1 Executive Summary

The purpose of this document is to provide information on the characteristics and overall functionality of a reference implementation for an Enterprise Wallet that aligns with the EBSI identity model, enabling the issuance and verification of verifiable credentials. This reference implementation offers European stakeholders a valuable foundation for managing verifiable credentials consistently with EBSI standards. By adopting this standardized approach, organizations can streamline development processes, reduce associated costs, and ensure seamless integration with EBSI's trusted digital infrastructure.

This reference implementation is designed to serve as a starting point for new Wallets aiming to adopt the EBSI identity model. Likewise, it can also **be utilized by existing Wallets for interoperability exercises or as a reference for features that may not yet be fully established.** Additionally, the Wallet is intended to act as **an initial point for the emergence of new discussions** and debates on **general or specific improvement proposals that could enhance the underlying quality of future or current solutions**.

After submitting and evaluating this deliverable, the EBSI-VECTOR project will engage in an extensive exercise to improve the interoperability of enterprise wallets and bring the EBSI conformance test to a higher level. This work will involve all the wallet providers in the consortium in a collaborative effort to create several interoperable reference implementations and will be submitted as an annex to this deliverable.

This implementation has been developed and conceived with the primary intention of complying with EBSI's specifications for issuance and verification, and, consequently, with the versions of the OID4VCI [1] and OID4VP [2] RFCs upon which it is based; and with the secondary intention of supporting the pilot scenarios of this project that lack a technical partner. To this end, one of the main goals of the implementation has been to pass EBSI's conformance tests while also proposing a model for integrating the Wallet with data storage systems for its users. The Wallet is defined as use-case agnostic and, consequently, does not have the capability to store data related to the underlying business logic. As a result, it requires integrations with other systems to retrieve the information needed to complete the issuance and verification processes. In the first case, it retrieves the data needed to complete the credential, and in the second, it validates the information provided by the user against the business logic's known data. Additionally, the Wallet

Project co-funded by the European Union under the Digital Europe Programme. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union. Neither the European Union nor the granting authority can be held responsible for them.

includes some extra features that future implementers may also benefit from, most notably the ability to revoke issued credentials.

Project co-funded by the European Union under the Digital Europe Programme. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union. Neither the European Union nor the granting authority can be held responsible for them.

2 Introduction

This deliverable introduces a reference implementation for an Enterprise Wallet that aligns with EBSI's identity model, aimed at supporting the issuance, verification, and revocation of verifiable credentials. This implementation serves as a foundational tool for stakeholders, streamlining the development and integration of credential management solutions while promoting interoperability and adherence to EBSI standards. Additionally, this initial version of the reference implementation is intended to serve as a starting point for future developments aimed at improving the existing solution and meeting the needs of the use cases characterizing WP4 and WP5 of EBSI VECTOR.

The purpose of this document is twofold: to establish a reliable starting point for Enterprise Wallet developments and to propose integration models that enable seamless credential operations through external data systems. Each chapter of this document explores a different facet of the implementation: initial chapters focus on the architecture and data model. Further sections delve into the specifics of issuance and verification flows, the role of integrations for credential data retrieval, and additional capabilities such as credential revocation.

Inspired by this reference implementation the different wallet providers in the consortium will start a collaborative effort to ensure more interoperability between the different solutions to create a sustainable and healthy wallet market in the EBSI and EUDI context. This work will be reported in a later annex to this deliverable.

Project co-funded by the European Union under the Digital Europe Programme. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union. Neither the European Union nor the granting authority can be held responsible for them.

3 License

The reference implementation of the Enterprise Wallet is offered under a dual-license model, providing flexibility to meet a variety of user needs. Users may choose between the AGPL-3.0 license, which adheres to open-source principles, and a commercial license option available through a private agreement with Izertis.

Under the AGPL-3.0 license, users are granted the freedom to use, modify, and distribute the software, even for commercial purposes. Any modifications distributed under this license must also comply with AGPL-3.0, thereby ensuring that all derivative works remain open source. This option is ideal for organizations committed to contributing to an open ecosystem and benefiting from community-driven improvements to the Enterprise Wallet.

Alternatively, the commercial license provides greater flexibility, allowing modifications to the software without the requirement to distribute or publish these changes. This option suits organizations with proprietary requirements or those seeking to integrate the Wallet within a closed environment without open-source obligations.

Further details, including licensing terms and conditions, are available in the official repository of the reference implementation, as referenced in the “Resources” section of this document.

Project co-funded by the European Union under the Digital Europe Programme. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union. Neither the European Union nor the granting authority can be held responsible for them.

4 Wallet Capabilities

This section provides an overview of the main capabilities of the Enterprise Wallet reference implementation. The tables below summarize essential features, such as credential formats, issuance and verification flows and revocation methods.

Verifiable Credentials	
Format	JWT_VC
Data Model	W3C Verifiable Credentials Data Model v1 [3]

Table 1: Verifiable Credentials Characteristics

Verifiable Credentials Issuance	
RFC Version	OID4VCI v10
Supported Flows	InTime, Deferred, PreAuthorize, Deferred and PreAuthorize combined.
Authorization method	ID Token, VP Token
Others	Notification of VC ID when they are successfully issued to a Holder Wallet.

Table 2: Issuance Capabilities

Verifiable Presentation Verification	
RFC Version	OIDVP v13
Supported Flows	Independent verification process or integrated in verifiable credential issuance.
Format	JWT_VP
Validations	• VP signature • VCs signatures • VC expiration date • VC “not before” date • VC status (revoked or not) • VC Terms of use (Trust chain) • VC Schema validation • VC claims external validation according to business logic.

Project co-funded by the European Union under the Digital Europe Programme. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union. Neither the European Union nor the granting authority can be held responsible for them.

Table 3: Verifiable Presentation Verification

Verifiable Credential Revocation	
Strategy	Verifier asks Issuer for status information
Data Structure	StatusList2021
Format	JWT_VP

Table 4: Verifiable Credential Revocation

Interfaces	
GUI	Administration interface for the Wallet that can be used for configuration purposes.
API	REST API conformance with EBSI guidelines. REST API for revocation and generation of QR codes

Table 5: Interfaces

Project co-funded by the European Union under the Digital Europe Programme. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union. Neither the European Union nor the granting authority can be held responsible for them.

5 Architecture

This section outlines the architecture of the Enterprise Wallet reference implementation. The following tables provide details on each architectural component, including their roles, interactions, and technologies

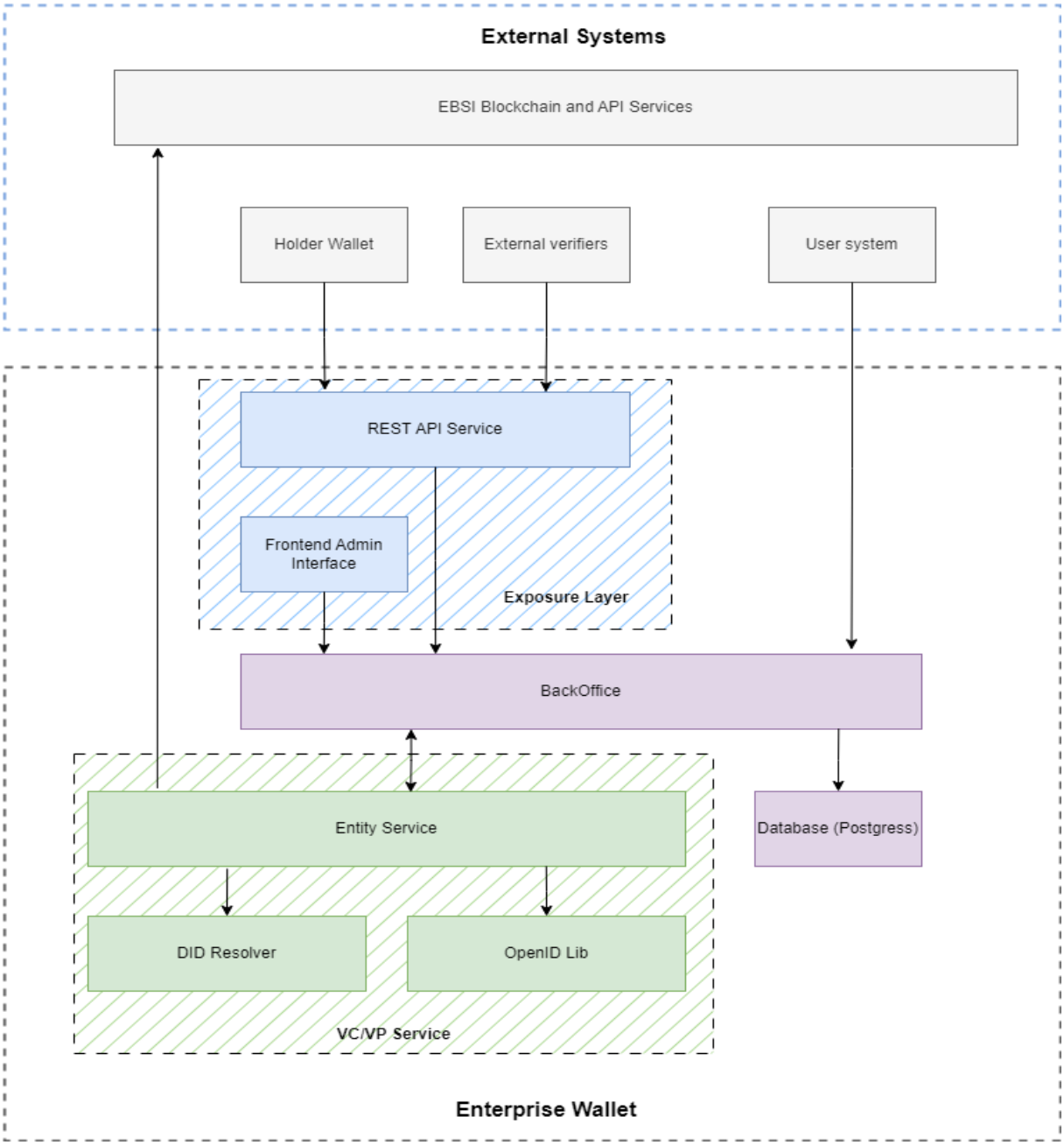

Figure 1: Architecture

Project co-funded by the European Union under the Digital Europe Programme. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union. Neither the European Union nor the granting authority can be held responsible for them.

The previous image presents a high-level diagram of the architecture of the reference implementation. As can be seen, the Wallet itself is composed of two major functional components: a BackOffice that handles exposing any means of interaction with the Wallet, and a service that uses a library to handle issuance and verification requests. Each of these components will be explained in detail below.

Component	**BackOffice**
Belongs to	Enterprise Wallet
Technological Stack	Python 3 Django
Functionality	
This is the central component of the Enterprise Wallet, responsible for offering user interfaces and APIs that will be used by other services and external components. Similarly, it is the only component with access to the underlying database, and therefore manages user handling logic and defines the issuance/verification flows, in addition to defining the necessary data models. Relations with other components:	
Relations with other components	
Front End Admin Interface	The BackOffice defines the administration models shown via this Front End. Consequently, the BackOffice is responsible for determining which data can be displayed and in what formats.
REST API Service	The BackOffice handles all API requests. If the requests relate to credential issuance, verification, or revocation, the BackOffice forwards them to the "Entity Service."
Entity Service	This service is not exposed externally due to its stateless nature and high level of integration with the BackOffice. Consequently, the BackOffice receives all external user requests and communicates them to the service when needed. Additionally, it provides several endpoints that the service can use to retrieve necessary information for issuance and verification processes, such as the schema to be used for a credential.
Database	The BackOffice is the only component with access to the database. Any other component that wants to access the information must do so

Project co-funded by the European Union under the Digital Europe Programme. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union. Neither the European Union nor the granting authority can be held responsible for them.

	through the API exposed by the BackOffice. Additionally, the BackOffice defines all the underlying data models required for the proper functioning of the solution.
User System	This component is also primarily responsible for ensuring the Wallet's integration with its users' data systems. When accessing information or sending notifications is required, the BackOffice handles the request. This mainly implies that the BackOffice does not store sensitive credential data but retrieves it as needed and discards it after generating the credential.

Table 6: Architecture - BackOffice

Component	**Database (Postgress)**
Belongs to	Enterprise Wallet
Technological Stack	Postgres DB
Functionality	
This is the underlying database for the Wallet. It stores the following information: • Definitions of issuance flows (one for each type of credential to be issued). • Definitions of verification flows (one for each independent verification process to be supported). • Information mandatory for integrations with the user's external services (a URL that must meet certain endpoints). • History of successfully issued credentials (ID + Date + DID Holder). Subject data is not stored. • User and permission management. • Cryptographic material to be used. • DID to be used.	
Relations with other components	

Table 7: Architecture - Database

Component	**REST API Service**
Belongs to	Exposure Layer – Enterprise Wallet

Project co-funded by the European Union under the Digital Europe Programme. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union. Neither the European Union nor the granting authority can be held responsible for them.

Technological Stack	REST API
Functionality	
A REST API that exposes all the endpoints for interacting with the Enterprise Wallet. These endpoints can be grouped into six major categories: • Endpoints related to OID4VP **[2]** and OID4VCI **[1]**: The API includes all the necessary calls to successfully issue or verify a credential. These calls are mainly managed by the "Entity Service," but the BackOffice acts as an intermediary between the user and the service. • Endpoints related to user management: The API offers several endpoints for user authentication and password recovery. The first function can be used to obtain access tokens other than those obtainable with OpenID, which are required for other operations. • Endpoints to retrieve information on issuance/verification flows: These endpoints are specifically designed to be used by the "Entity Service" and allow retrieval of configuration information for issuance and verification flows (e.g., the schema to be used or the type of token to request from the user for authentication). These endpoints require an access token. • Endpoints for integration with client applications: A series of endpoints are provided for client applications to integrate issuance, verification, and revocation processes with their systems. This includes endpoints for generating QRs codes to start these processes and another to request credential revocation. • Endpoints related to nonce management: Since the service does not have access to the database, it relies on the BackOffice for any operation requiring reading or writing to the database, including nonce management. • Endpoint to check revocation information: An endpoint is exposed to allow external verifiers to obtain Status VC to check the state of credentials issued by the Enterprise Wallet.	
Relations with other components	
BackOffice	All API requests are handled by the BackOffice.

Table 8: Architecture - REST API

Project co-funded by the European Union under the Digital Europe Programme. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union. Neither the European Union nor the granting authority can be held responsible for them.

Component	**FrontEnd Admin Interface**
Belongs to	Exposure Layer – Enterprise Wallet
Technological Stack	Python 3, Django
Functionality	
An admin interface that allows Enterprise Wallet users to adjust configuration parameters and define the issuance and verification processes they wish to implement. The admin interface does not expose any forms to fill in credential data. This data must be provided through the Wallet's integration with user services (User Systems) as outlined in the architecture image. Among the configuration parameters, the option to define the service URL that provides this information to the Wallet and acts as the Authentic Source is included.	
Relations with other components	
BackOffice	The graphical interface is directly provided by the BackOffice, which defines each and every administrative component.

Table 9: Architecture - FrontEnd Admin Interface

Component	**Entity Service**
Belongs to	VC/VP Service – Enterprise Wallet
Technological Stack	NodeJS
Functionality	
This is a stateless service responsible for managing OpenID processes, both for credential issuance and verification. It also generates status credentials used to indicate the revocation of associated credentials.	
Relations with other components	
BackOffice	The service retrieves all necessary information from the BackOffice via the REST API exposed by it. It obtains configuration information for the supported issuance and verification flows. Additionally, any nonce or other information that needs to be stored between OpenID stages is also

Project co-funded by the European Union under the Digital Europe Programme. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union. Neither the European Union nor the granting authority can be held responsible for them.

	sent to the BackOffice, where it is temporarily stored, usually associated with the nonce itself.
DID Resolver	The service uses this component's functionalities to resolve the DID Documents of the various DIDs involved in an issuance and/or verification process.
OpenID Lib	In actual case, the service does not implement OpenID itself but uses this library, which contains all the associated logic. This library verifies authorization/authentication requests, validates VP, and generates verifiable credentials. The main function of the service, therefore, is to obtain and provide the necessary information to use the library's methods

Table 10: Architecture - Entity Service

Component	**DID Resolver**
Belongs to	VC/VP Service – Enterprise Wallet
Technological Stack	NodeJS
Functionality	
An internal component of the "Entity Service" responsible for resolving DID Documents based on a DID.	
Relations with other components	
Entity Service	The Entity Service uses this component to resolve DID Documents [4]
EBSI Blockchain and API Services	When operating with EBSI DIDs [5], for example, to verify a credential signature in a verification process, the EBSI DID Registry is consulted to obtain the associated DID Document.

Table 11: Architecture - DID Resolver

Component	**OpenID Lib**
Belongs to	VC/VP Service – Enterprise Wallet
Technological Stack	NodeJS

Project co-funded by the European Union under the Digital Europe Programme. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union. Neither the European Union nor the granting authority can be held responsible for them.

Functionality	
A library that implements OID4VCI **[1]** and OID4VP **[2]**. It has the following features: • Resolution of ID Tokens and VP Tokens. • Generation of verifiable credentials according to the W3C data model, both version 1 **[3]** and 2 **[6]**. • Support for deferred and pre-authorized issuance flows.	
Relations with other components	
Entity Service	The Entity Service uses this library to resolve the authorization, authentication, and issuance/verification requests it receives from the BackOffice, which in turn are requested by a Holder Wallet

Table 12:Architecture - OpenID Lib

Project co-funded by the European Union under the Digital Europe Programme. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union. Neither the European Union nor the granting authority can be held responsible for them.

6 Data Model

This section presents the data model of the Enterprise Wallet reference implementation. It describes the key data structures, their purposes, and their roles in supporting the issuance and verification processes.

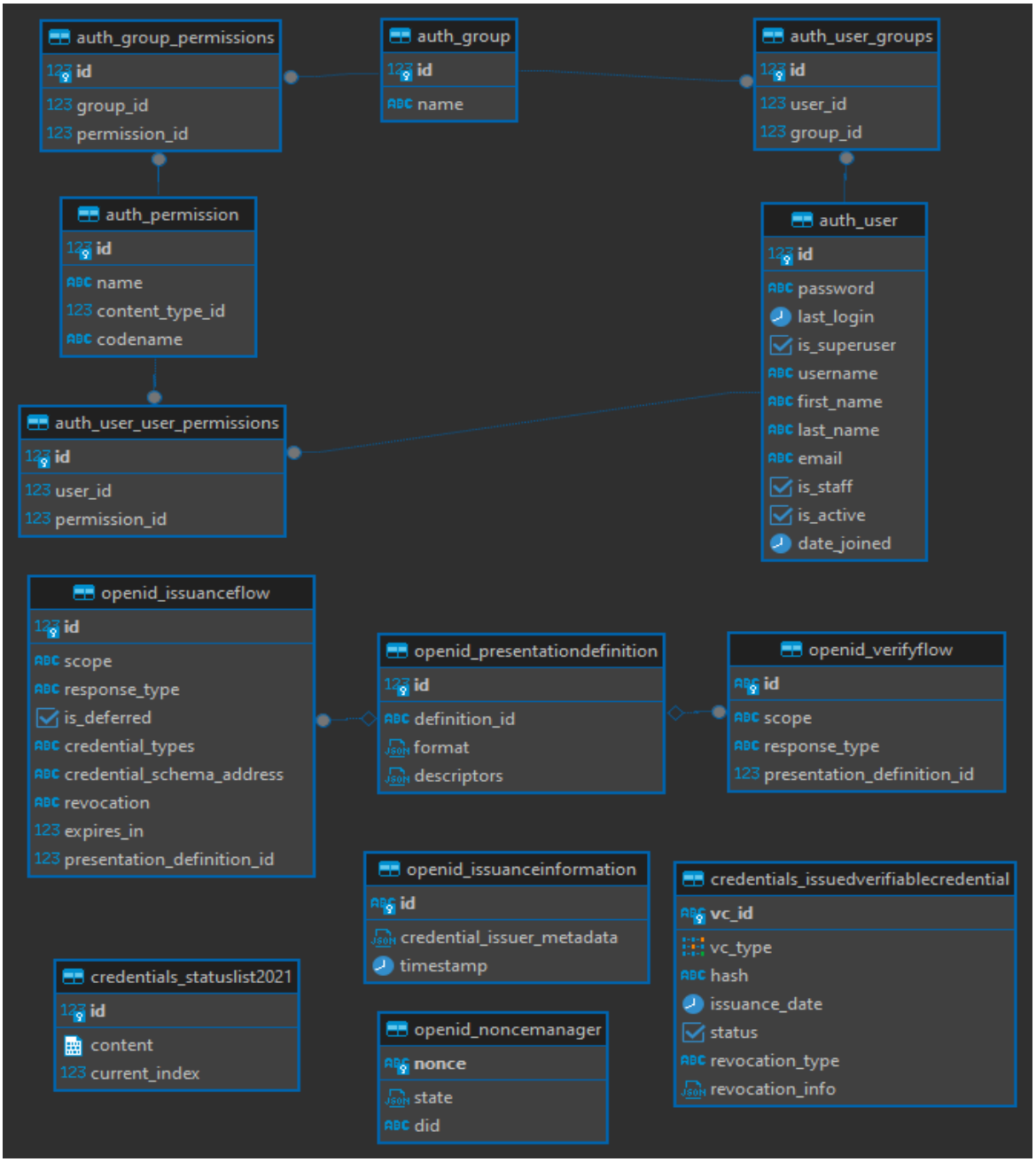

Figure 2: Data Model

Project co-funded by the European Union under the Digital Europe Programme. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union. Neither the European Union nor the granting authority can be held responsible for them.

In the previous image, you can see the relationships between the existing tables in the database. The six upper collections (*auth_group_permissions, auth_group, auth_user_permissions, auth_user, auth_group,* and *auth_user_groups*) are all related to the management of users and their corresponding permissions. Each user is assigned a series of permissions that can either be individual or provided by a predefined group, which the Wallet's superusers can modify at their discretion. Each user is associated with a username and password, necessary to log into the system. These data collections can only be modified via the admin interface and only by users who already possess the required permissions. Along with this document, an additional resource is provided that explains in more detail how to carry out these operations.

The seven lower tables, in turn, define the data structures that contain information on the issuance and verification flows and will be explained in detail below:

6.1 Presentation Definition

This table contains the specification of the different presentation definitions, according to DIF [7], that can be used with the Wallet when requesting VP Tokens from the Holder Wallets. The model requires specifying the definition identifier, which must be respected when constructing the Presentation Submission, as well as the format and the definition body, both corresponding to those from the DIF. The user can modify this collection through the admin interface.

6.2 Issuance Flow

This collection contains the defined issuance flows supported by the Enterprise Wallet. A flow must be defined for each type of credential to be issued. As seen in the diagram, the user must specify the following:

- **Scope**: The scope to be used for this issuance flow. Default is *openid*.
- **Response type**: The type of token (ID or VP) that will be requested from the user in InTime and deferred flows.
- **Is Deferred**: Indicates whether the issuance should be deferred or not.
- **Credential Types**: The specific type of the credential. It is not mandatory to include *VerifiableCredential* and *VerifiableAttestation*.
- **Credential Schema Address**: URI where the credential schema is hosted.
- **Presentation Definition ID**: Presentation Definition to be used when requesting a VP Token from the Holder Wallet.

Project co-funded by the European Union under the Digital Europe Programme. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union. Neither the European Union nor the granting authority can be held responsible for them.

- **Revocation**: Revocation strategy to be used. This is optional, and the only existing possibility is StatusList2021 [8].
- **Expires In**: Allows specifying after how many seconds the credential should expire. Useful for short-lived credentials. If no value is indicated, issued credentials will not expire.

Users can interact with this model using the admin interface.

6.3 Verify Flow

This collection contains the definition of independent verification flows, meaning flows that do not result in credential issuance. One entry must be defined in the table for each flow to be created.

- **Scope**: The scope of the flow, which also serves as its identifier.
- **Response Type**: The type of token that will be requested from the user during the verification flow (ID Token or VP Token). In most cases, a VP Token will be used, but the other option is required to pass the EBSI conformance tests.
- **Presentation Definition ID**: If a VP Token is used, this parameter must indicate which definition will be used.

Users can interact with this model using the admin interface.

6.4 Issuance Information

This table contains the metadata of the Issuer represented by the Enterprise Wallet in accordance with the EBSI specification [9]. The user only needs to generate one entry in the table, and it will automatically update whenever changes are made to the issuance flows. Changes to the verification flows have no impact on it.

Users can interact with this model using the admin interface. No issuance flow can begin, unless an instance of this model has been created. This restriction also applies to generating QRs with the API, which also depends on it.

6.5 Issued Verifiable Credentials

This table contains a history of all credentials successfully issued by the Enterprise Wallet. For each credential, only the holder's DID, the issuance date, and its identifier are stored. No credential subject data is ever saved in the system. Additionally, the model has a boolean

Project co-funded by the European Union under the Digital Europe Programme. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union. Neither the European Union nor the granting authority can be held responsible for them.

attribute indicating the credential's status, i.e., whether it is revoked or not. Furthermore, if the credential allows revocation, the relevant revocation information will also appear in the data entry.

6.6 StatusList2021

Each entry in this table corresponds to an instance of a binary list that allows expressing the status of numerous credentials. Specifically, each issued credential is assigned an index in the list, and if its value changes from 0 to 1, the credential is assumed to be revoked. All issued credentials indicate in the "CredentialStatus" field information for retrieving the associated list and the index to be checked within it. Along with the list, the table stores an index indicating the next unassigned index. Each list has a maximum size of 16 KB, implying a total of 131,072 possible records. If this limit is reached, another entry is added to the table.

6.7 Nonce Manager

This table stores the "nonces" used by the "Entity Service". As it lacks its own database, the service requires an external or additional service to store information necessary for completing an issuance and/or verification flow. In practice, this table could be managed by an additional service, as it has no impact on the rest. Along with each "nonce," the DID of the person who received it is stored, as well as a status that allows the service to store relevant information to consider for the OpenID phase in which it is, and which can be associated with the nonce.

Project co-funded by the European Union under the Digital Europe Programme. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union. Neither the European Union nor the granting authority can be held responsible for them.

7 Explanation of Core Functionalities

This chapter provides a detailed examination of the core functionalities and components within the Enterprise Wallet reference implementation. Each section explores a specific aspect of the system, including cryptographic storage, data management, and the flows for credential issuance and verification

7.1 Cryptographic Storage

The reference implementation can only work with one cryptographic material, which is kept in memory during the execution of the Wallet and is loaded through an environment variable. The key must be specified in JWK format, and its algorithm must be ES256. Since only one key is supported, if a change is desired, it is necessary to restart the Wallet and modify the corresponding environment variable.

All information about environment variables and configuration can be found in the component documentation in their respective repositories.

7.2 Data Management

The Enterprise Wallet is devoid to store information related to business logic and, as a result, does not provide any resource and/or utility that can be used to upload information to it, which could later be used to generate verifiable credentials. Instead, the Wallet is designed to integrate with existing data management systems. For this purpose, the Wallet requires a URL to be provided where the service it needs to integrate with is hosted. The expected interface for achieving this integration is detailed in depth in section 8 of this document. This URL, which is provided to the Wallet via the environment variable “ENTITY_URL”, is used for the following operations:

- Retrieve user data to include in credentials.
- Notify the user about deferred issuance to obtain an acceptance token.
- Exchange an acceptance token for credential data in the deferred flow.
- Notify the user about a successfully issued credential.
- Request additional verification of the verifiable presentation data according to the business logic.

Project co-funded by the European Union under the Digital Europe Programme. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union. Neither the European Union nor the granting authority can be held responsible for them.

This design decision ensures that the Wallet never holds sensitive user information, reducing interactions with the user to the final stages of the issuance process (signing the credential once the data is obtained), after which it is removed from memory.

If a service to provide the data is not available, a flag can be activated via an environment variable that allows bypassing this restriction. In this case, the Wallet will generate credentials, but without data. This variable should only be considered for tests where the content of the credentials is not relevant, but rather the execution of the flow itself. For more information, check the configuration documentation associated with the BackEnd component repository.

7.3 Issuance Flow Definition

The Wallet supports four different issuance flows: InTime, deferred, pre-authorized, and a combination of the latter two. In any case, any credential issuance process must be predefined in the system and stored in the database in the "Issuance Flow" table. For this purpose, users can use the admin interface to add instances to this table. In these instances, the type of credential and its schema must be specified, along with the type of authorization requested from the user and whether to support revocation. Below is how to define each type of flow.

- **InTime Flow**: All defined issuance processes are InTime by default.
- **Deferred Flow**: To specify that a credential should be issued using this flow, the "Is Deferred" boolean in the instance must be set to true. Once set, all credentials will use this flow, making it mutually exclusive with the InTime flow.
- **Pre-Authorized Flow**: All defined flows always support pre-authorized issuance, regardless of their characteristics. However, its definition is not carried out in the model but is instead delegated to the generation of the QR codes. When a QR code is requested for credential issuance, an optional parameter corresponding to the pre-authorized code can be indicated. If provided, this will be included in the Credential Offer, and a Holder Wallet will be able to use it for authentication.
- **Pre-Authorized + Deferred Flow**: Define a flow as deferred and generate a Credential Offer QR with a pre-authorized code as indicated in the previous section.

Regarding credential data or resolving the codes/tokens involved in different flows, the pertinent integrations must be provided, as mentioned previously in section 6.2.

Project co-funded by the European Union under the Digital Europe Programme. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union. Neither the European Union nor the granting authority can be held responsible for them.

Once at least one issuance flow has been defined, an "Issuance Information" entry must be generated, a model that allows the generation of the necessary QRs and links for Cross Device and Same Device issuance.

7.4 Issuance Notifications

Each time a credential is successfully issued, the Wallet sends a notification to the data provider with certain details of the issued credential, specifically the DID of its recipient, its issuance date and "not before", its type, and, most importantly, its identifier. This information can be stored by the data provider to maintain a parallel issuance history, but its primary purpose is to inform about parameters that may be necessary for requesting the subsequent revocation of the credential. The Wallet allows credential revocation both through the admin interface and via an endpoint, but in both cases, the credential to be revoked must be indicated by its ID. If a more complex revocation logic is desired, such as revoking all credentials issued on certain days, the integrator must implement this logic using the data from these notifications.

More information about these notifications can be found in the "Integrations" section.

7.5 Verification Flow Definition

The Wallet allows defining independent verification flows without issuing credentials. To do this, it is necessary to add entries to the "Verify Flow" table, an action that can be performed through the admin interface. Each verification process is differentiated from others by its "scope," which acts as a unique identifier. Verification processes require the user to submit a token for authentication, with the two available options being "ID Token" and "VP Token". The first verifies that the user has control over a DID, while the second, in addition to the aforementioned, allows verifying additional information in the form of the subject's "claims", for example, their age or academic degree, which the user provides through verifiable credentials. For this purpose, VP Tokens require a Presentation Definition, which allows specifying the requirements.

7.6 Credential Revocation

The revocation of a credential is a process that can only be performed if previously issued credentials indicated revocation support in their respective issuance flow (sections 5.2 and 6.3) and only through the admin interface or via an endpoint:

Project co-funded by the European Union under the Digital Europe Programme. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union. Neither the European Union nor the granting authority can be held responsible for them.

- Via the Admin Interface: The "Issued Verifiable Credentials" entry for the issued credential must be searched, and then the boolean attribute "Status" must be marked. After that, the system will automatically update the credential status, and any subsequent verification will reflect its revoked status.
- Via Endpoint: The Wallet's REST API exposes an endpoint that allows revoking credentials by their ID, an attribute notified to the integrator as described in section 6.4. This method does not support revocation by other parameters or group revocations. Revoking a credential with this method also updates the corresponding entry in the "Issued Verifiable Credential" table.

Regardless of the issued credential, the structure used to reflect their status is StatusList2021. This is a binary list where each credential is assigned a bit. If the bit is updated to 1, the credential is considered revoked. This system, due to its binary nature, does not allow additional statuses like suspension. It also does not allow indicating the reason for revocation. In theory, more than one status could be allowed by using more than one list, but the Wallet's efforts have focused solely on one.

Regarding the revocation strategy, verifiers aiming to check the status of credentials must consult directly with the issuers, without using any proxy as an intermediary. This approach was chosen for its relative simplicity compared to other methods. If EBSI were used as a proxy, it would first be necessary to register the corresponding DID as a Trusted Issuer and then register an associated proxy, which prevents testing with alternative DIDs. Similarly, the reference implementation does not include logic for completing the onboarding processes in EBSI, so adding this restriction would require the user to use additional tools.

7.7 Cross Device and Same-Device Support

The Wallet supports issuing credentials both Cross Device and Same Device format. For each case, a specific endpoint is provided that allows, on the one hand, the generation of a QR code and, on the other, a deep link. The process of both generates the same Credential Offer with all the credentials the Wallet can issue, essentially one for each "Issuance Flow" defined in the database. Although the entire API is available in the “Resources” section of this document, the endpoints in question will be outlined below:

Project co-funded by the European Union under the Digital Europe Programme. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union. Neither the European Union nor the granting authority can be held responsible for them.

```
{
  "/credential-offer/qr": {
    "get": {
      "operationId": "credential-offer_get_credential_offer_cross_device_by_issuer",
      "description": "GET Credential Offer QR",
      "parameters": [
        {
          "name": "pre-authorized_code",
          "in": "query",
          "description": "Preauthorize code",
          "type": "string"
        },
        {
          "name": "user_pin_required",
          "in": "query",
          "description": "User pin required",
          "type": "boolean"
        }
      ],
      "responses": {
        "200": {
          "description": "",
          "schema": {
            "$ref": "#/definitions/Qr"
          }
        }
      },
      "tags": [
        "credential-offer"
      ]
    },
    "parameters": [
      {
        "name": "id",
        "in": "path",
        "required": true,
        "type": "string"
      }
    ]
  }
}
```



```
{
  "/credential-offer/url": {
    "get": {
      "operationId": "credential-offer_get_credential_offer_same_device_by_issuer",
      "description": "GET Credential Offer",
```


Project co-funded by the European Union under the Digital Europe Programme. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union. Neither the European Union nor the granting authority can be held responsible for them.

```
      "parameters": [
        {
          "name": "pre-authorized_code",
          "in": "query",
          "description": "Preauthorize code",
          "type": "string"
        },
        {
          "name": "user_pin_required",
          "in": "query",
          "description": "User pin required",
          "type": "boolean"
        }
      ],
      "responses": {
        "200": {
          "description": ""
        },
        "302": {
          "description": "",
          "schema": {
            "$ref": "#/definitions/IssuanceOfferResponse"
          }
        }
      },
      "tags": [
        "credential-offer"
      ]
    },
    "parameters": [
      {
        "name": "id",
        "in": "path",
        "required": true,
        "type": "string"
      }
    ]
  }
}
```

As can be seen, both endpoints are very similar, as their purpose is the same: to provide a method for obtaining "Credential Offers". Each parameter involved will be detailed:

- **pre-authorized_code**: Allows specifying a pre-authorized code to include in the Credential Offer, so that the Holder Wallet can use it in its credential request. This parameter is optional.

Project co-funded by the European Union under the Digital Europe Programme. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union. Neither the European Union nor the granting authority can be held responsible for them.

- **user_pin_required**: Indicates whether, during the execution of the deferred flow, the user should provide a PIN. This parameter is optional, and its default value is "false".

One feature of this implementation that may differentiate it from the rest is the approach taken regarding the verification processes. In order to add value to the proposal, the Wallet allows verification processes to be carried out via QR. These QRs codes do not provide any data structure similar to a "Credential Offer", but instead correspond directly to a VP Token Request issued by the Wallet, specific to a particular verification process. In other words, they contain the Presentation Definition that the Holder Wallet must satisfy. After processing the QR, the Holder Wallet can respond directly with a VP.

A problem that arises with this proposal is how to link the verification process to a real resource, such that a successful verification unlocks access to it. To address this, it has been decided to use the "state" field of authorization requests, with a VP Token Request being one of them. This state must be generated by the client application and indicated to the Wallet when requesting the QR code generation. This will cause this parameter to be included in the corresponding request, and as a result, the Holder Wallet can include it in its response. After verifying the delivered VP, the Wallet will use the verification process integration to communicate both the data extracted from the VP and the state associated with it. The portal, upon receiving this information, must validate the data according to its business logic and, if correct, can use the "state" to identify the process to be unlocked on its side.

Below are both the endpoint in question that allows the generation of these QRs codes and a diagram exemplifying the described process.

Project co-funded by the European Union under the Digital Europe Programme. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union. Neither the European Union nor the granting authority can be held responsible for them.

```
{
"/presentation-offer-request/qr": {
      "get": {
        "operationId": "presentation-offer-
request_get_presentation_offer_cross_device_by_verifier",
        "description": "GET Presentation Offer QR",
        "parameters": [
          {
            "name": "verify_flow",
            "in": "query",
            "description": "Verifiable Presentation Scope",
            "type": "string"
          },
          {
            "name": "state",
            "in": "query",
            "description": "State code",
            "required": false,
            "type": "string"
          }
        ],
        "responses": {
          "200": {
            "description": "",
            "schema": {
              "$ref": "#/definitions/Qr"
            }
          }
        },
        "tags": [
          "presentation-offer-request"
        ]
```


Project co-funded by the European Union under the Digital Europe Programme. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union. Neither the European Union nor the granting authority can be held responsible for them.

```
      },
      "parameters": []
    },
    "/presentation-offer-request/url": {
      "get": {
        "operationId": "presentation-offer-
request_get_presentation_offer_same_device_by_verifier",
        "description": "GET Presentation Offer",
        "parameters": [
          {
            "name": "verify_flow",
            "in": "query",
            "description": "Verifiable Presentation Scope",
            "type": "string"
          },
          {
            "name": "state",
            "in": "query",
            "description": "State code",
            "required": false,
            "type": "string"
          }
        ],
        "responses": {
          "200": {
            "description": ""
          },
          "302": {
            "description": "",
            "schema": {
              "$ref": "#/definitions/PresentationResponse"
            }
```


Project co-funded by the European Union under the Digital Europe Programme. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union. Neither the European Union nor the granting authority can be held responsible for them.

```
          }
        },
        "tags": [
          "presentation-offer-request"
        ]
      },
      "parameters": []
    }

}
```


Project co-funded by the European Union under the Digital Europe Programme. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union. Neither the European Union nor the granting authority can be held responsible for them.

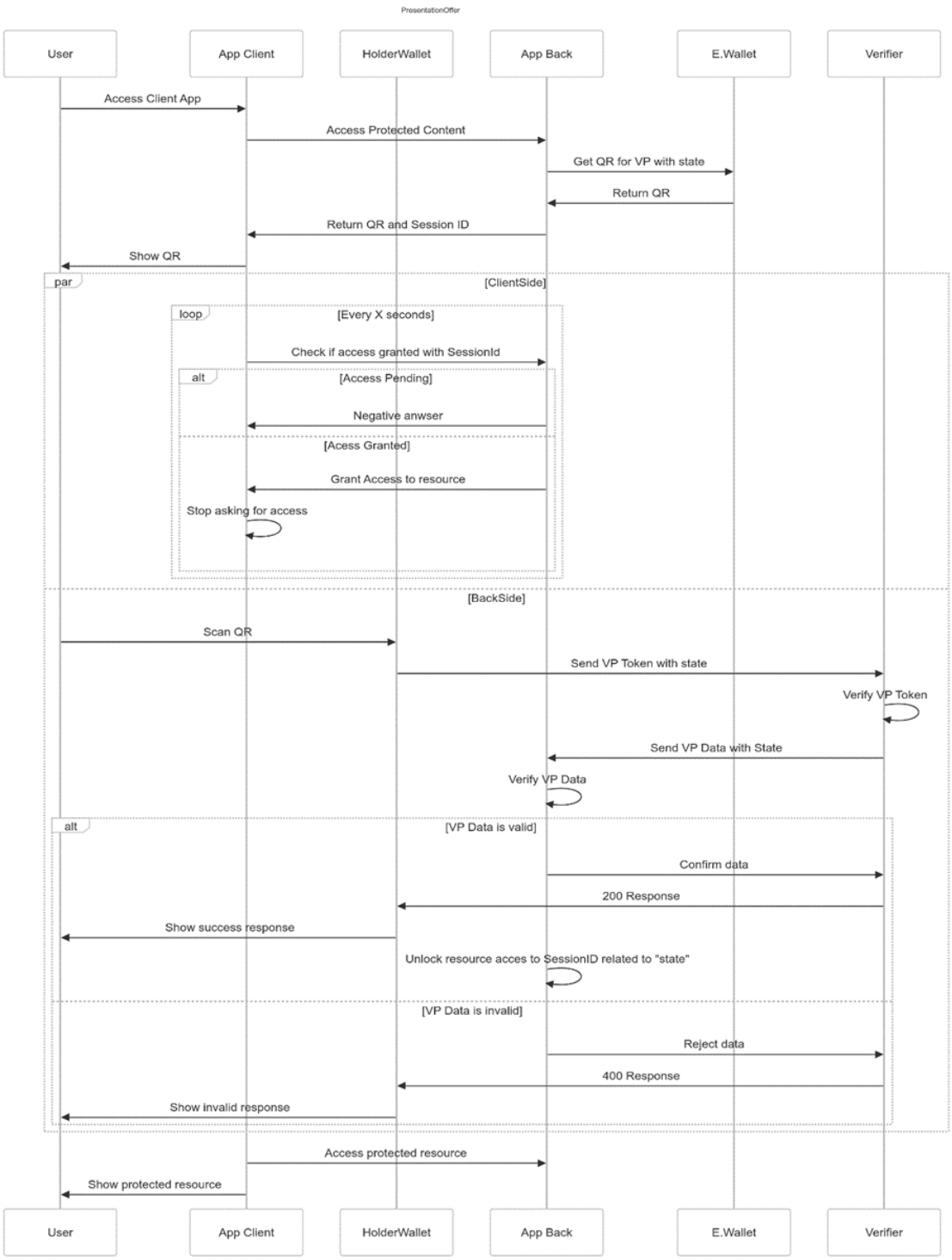

Figure 3: Verification Example Diagram

Project co-funded by the European Union under the Digital Europe Programme. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union. Neither the European Union nor the granting authority can be held responsible for them.

8 Integrations

The Enterprise Wallet requires the Authentic Source to enable several endpoints to achieve the necessary integrations for the issuance, verification, and revocation of credentials. This section will detail each of them, highlighting their peculiarities and the scenarios in which they are involved. It is important to note that it is not necessary to support all integrations, only those that are needed for the use case.

8.1 Integrations for credential Issuance

8.1.1 InTime Issuance

In the InTime flow, the credential is delivered to the user at the moment of their request, and OpenID is used for user authentication. This flow is simple to implement but only allows the user to be identified by their DID, as no additional identifier is involved during the flow.

For this flow, the Authentic Source must provide the Enterprise Wallet with a single endpoint:

```
openapi: 3.0.3
info:
  title: Credential Data Integration
  description: |-
    Endpoint specification for Credential Data integration
  version: 1.0.0
paths:
  /credentials/external-data:
    get:
      description: This endpoint will be called by the Enterprise Wallet, and it must
provide the subject data of the verifiable credential as a JSON object.
      parameters:
        - name: vc_type
          description: The VC type
          in: query
          schema:
            type: string
          required: true
        - name: user_id
          description: The user ID. It can be a DID or a PreAuthzCode
          in: query
          schema:
            type: string
          required: true
        - name: pin
          description: PIN given by the user. Only for PreAuthzFlows
```


Project co-funded by the European Union under the Digital Europe Programme. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union. Neither the European Union nor the granting authority can be held responsible for them.

```
        in: query
        schema:
          type: string
        required: false
    responses:
      '200':
        description: Successful response with VC Data
        content:
           application/json:
            schema:
              type: object
```

As can be seen, this is a GET method in which the Wallet sends two parameters. The first (vc_type) corresponds to the type of credential requested, while the second (user_id) is the user identifier, which in this issuance flow corresponds to the DID (in the authorized flow, discussed later, it is a token). The Authentic Source must be able to identify the user and generate/retrieve the associated data to include in the credential using only these two parameters. If this is not possible, other methods described in this section will need to be used.

Advantages	Disadvantages
Simple to implement	The user can only be identified by his DID.
It does not require any additional authentication system.	

Table 13: Integrations -InTime

8.1.2 Pre-Authorized Issuance

In this flow, authorization/authentication occurs in external processes and with protocols that may not be related to OpenID. Consequently, it is not required to know the user's DID to identify them. To adequately explain this flow, consider the following example:

A user accesses a web application that has its own authentication/authorization methods. The user identifies themselves and requests the issuance of a verifiable credential. The application accepts the request and generates a token that it provides to the user. This token can then be presented to the Enterprise Wallet responsible for generating the verifiable credential. Upon receiving it, the Wallet submits it to the Authentic Source to determine its validity. If valid, the

Project co-funded by the European Union under the Digital Europe Programme. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union. Neither the European Union nor the granting authority can be held responsible for them.

Authentic Source must respond with the credential data associated with that token and, consequently, the user for whom it was issued.

This flow requires the same integration as the indicated for the previous flow, but in this case, the user identifier will be the mentioned token instead of a DID.

Advantages	Disadvantages
Allows the use of alternative authorization/authentication methods.	The Authentic Source must have a method of authorization/authentication on its side.
It allows linking the authentication process to the business logic.	The Authentic Source must implement token management logic.

Table 14: Integrations - PreAuthorize

8.1.3 Deferred Issuance

Deferred issuance is identical to In-Time issuance, with the distinction that the credential is not issued at the end of the issuance flow; instead, the user is provided with an acceptance code that they can later exchange for the verifiable credential. This flow allows for the deferral of the credential issuance until a condition is met. For example, the credential data may not yet be available, or additional verification of the request by a human user may be necessary. In any case, when a request for such a credential is received, the Enterprise Wallet communicates it to the Authentic Source, which must respond with the acceptance code. Likewise, the Authentic Source must provide a second endpoint that allows for the exchange of that code for the credential data itself. Therefore, this flow requires two integrations: one to obtain the acceptance codes and another to redeem them.

```
/deferred/registry:
    post:
      description: This endpoint is used to get a new acceptance token used in the
deferred flow.
      requestBody:
        content:
          application/json:
            schema:
              type: object
              properties:
```


Project co-funded by the European Union under the Digital Europe Programme. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union. Neither the European Union nor the granting authority can be held responsible for them.

```
              vc_type:
                type: string
                description: The VC type
              client_id:
                type: string
                description: The user ID. It can be a DID or a PreAuthzCode
              pin:
                type: string
                description: PIN given by the user. Only for PreAuthFlows
            required:
              - vc_type
              - client_id
              - pin
      responses:
        '200':
          description: Successful response with acceptance code
          content:
             application/json:
              schema:
                type: string
```

```
/deferred/exchange/{code}:
    get:
      description: This endpoint is used to get a new acceptance token used in the
deferred flow.
      parameters:
        - name: code
          description: The acceptance code that can be used to get the credential
data
          in: path
          schema:
            type: string
          required: true
      responses:
        '200':
          description: Successful response with the credential data
          content:
             application/json:
              schema:
                oneOf:
                  - type: object
                    description: Credential data
                  - type: string
                    description: Acceptance token
```


Project co-funded by the European Union under the Digital Europe Programme. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union. Neither the European Union nor the granting authority can be held responsible for them.

Note that, in this last case, it is possible for the user to attempt to redeem the code before the credential is available. The Authentic Source must prepare for this possibility and implement countermeasures to respond appropriately. The simplest way is to respond to the attempt with a new acceptance code or simply return the same one.

Finally, it is important to highlight that the deferred flow can be combined with the pre-authorized flow. In this case, the user identifier provided in the registration endpoint changes from the user's DID to the token they have provided.

Advantages	Disadvantages
It allows to postpone the issuance of the credential according to the needs of the business logic.	Requires a greater number of integrations.
	The Authentic Source must manage the validity of the acceptance tokens.
	Ideally, the Authentic Source should implement a mechanism to notify the Holder when the credential is available for redemption.

Table 15: Integrations - Deferred

8.1.4 Notifications of issued credentials

Each time a credential is successfully issued, the Enterprise Wallet sends a notification to the Authentic Source with certain data about the mentioned credential that may be useful for its subsequent revocation. If the issuance fails for any reason, this message will not be sent.

```
/credentials/external-data:
    post:
      description: Notify about a verifiable credential that has been succesfull
issued.
      requestBody:
        content:
          application/json:
            schema:
              type: object
              properties:
                vc_type:
```


Project co-funded by the European Union under the Digital Europe Programme. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union. Neither the European Union nor the granting authority can be held responsible for them.

```
                  type: string
                  description: The VC type
                did:
                  type: string
                  description: The user DID.
                vc_id:
                  type: string
                  description: The ID of the verifiable credential
                nbf:
                  type: string
                  description: The validFrom date of the verifiable credential
                pre_code:
                  type: string
                  description: The PreAuthorization code used by the user. Only for
the pre-authorized flow
                issuance_date:
                  type: string
                  description: The issuance date of the verifiable credential
              required:
                - vc_type
                - did
                - vc_id
                - nbf
                - pre_code
                - issuance_date
      responses:
        '200':
          description: Successful response
```

As can be seen, information such as the type of credential, its identifier, the holder's DID, and the issuance date is sent, but never the credential itself or its user data. This information can be used, as mentioned at the beginning, to establish revocation methods of varying complexity. For example, a request could be made to revoke all credentials issued to a specific user or those issued on a particular day. Note that currently, the Wallet only supports revoking credentials using their identifier, so all these possibilities mentioned should be carried out by the applications of the Authentic Source.

It is important to highlight that if there is no intention to revoke credentials, then it is not necessary to support this endpoint.

Project co-funded by the European Union under the Digital Europe Programme. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union. Neither the European Union nor the granting authority can be held responsible for them.

8.2 Integrations for credential verification

Credential verification requires an additional endpoint where the Wallet will provide the data extracted from the credentials presented by the user via a verifiable presentation. The purpose of this presentation is double. On one hand, it communicates the extracted information that may be of interest to the Authentic Source, and on the other, it requests additional verification of the data according to business logic. Although the Wallet can verify if the content of a credential complies with a schema, it cannot discern whether the information is correct or not. For example, if the credential contains an identifier, only the Authentic Source can determine whether it exists or not.

The specific endpoint is as follows:

```
/presentations/external-data:
    post:
      description: Notify about a verifiable credential that has been successfully
issued.
      requestBody:
        content:
          application/json:
            schema:
              type: object
              properties:
                valid:
                  type: boolean
                  description: Flag that indicate if the verification was successful
or not.
                holderDid:
                  type: string
                  description: The user DID.
                state:
                  type: string
                  description: The state sent with the verifiable presentation
                claimsData:
                  type: object
                  description: The claims extracted from the verifiable presentation.
It is an empty object if valid is false
              required:
                - holderDid
                - valid
                - claimsData
      responses:
        '200':
          description: Successful response
```


Project co-funded by the European Union under the Digital Europe Programme. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union. Neither the European Union nor the granting authority can be held responsible for them.

As can be seen, this is a POST method in which the Wallet sends the DID of the user who presented the verifiable presentation and the extracted data from the credentials. Note that these are only a subset of all the information from the credential that meets the requirements of a presentation definition. For example, if the user was requested a credential with a date of birth, only that data will be delivered to the Authentic Source, not the rest, so it is important to construct the definitions properly. In addition to this information, the Wallet also sends a state. This corresponds to a parameter used when requesting a QR code for a verification process, linking the verification itself with access to a real resource on the Authentic Source side. Lastly, the "valid" field is a boolean indicating whether the verification was successful. If this parameter is "false", then the credential data will not be sent. The utility of this attribute is primarily informational, but when used with the "state," it allows for determining that access to the resource was denied, potentially activating corresponding measures if deemed necessary

8.3 Combination of Issuance and Verification Integrations to Achieve Identity Mapping

It should be noted that a verification process does not only occur independently; it can also be used as a method of authorization/authentication in issuance processes. This is critically important as it allows for issuance processes in which the user can be identified based on the data they presented with their credentials, negating the need to revert to their DID or the pre-authorized flow. If this method is to be used, support for the verification integration must be provided alongside the InTime issuance integration. From a functional perspective, both endpoints will continue to operate exactly the same and will send the corresponding information. However, since both are involved in the same issuance process, the Authentic Source can leverage this fact to perform identity mapping. Here's how it works:

The user is requested to provide a verifiable presentation containing a credential with an identifier known to the business logic. The user will submit the credential along with their DID to the Enterprise Wallet. The Wallet will communicate both elements to the Authentic Source, which must validate the provided data and also store the DID used by the user. Subsequently, the user will request a verifiable credential using that same DID. The Wallet will request credential data, indicating its type and the user's DID in the call. Since this DID is the same, the Authentic Source can link it to the previous verification process, allowing the identification of the

Project co-funded by the European Union under the Digital Europe Programme. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union. Neither the European Union nor the granting authority can be held responsible for them.

data to be included in the credential using the identifiers provided in the verification, rather than the DID

Advantages	Disadvantages
It allows the user to be identified using other verifiable credentials, which makes it possible to adjust it to numerous use cases.	Requires several integrations and more logic on the part of Authentic Source than other approaches.
Allows for identity mapping.	

Table 16: Integrations - Identity Mapping

Project co-funded by the European Union under the Digital Europe Programme. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union. Neither the European Union nor the granting authority can be held responsible for them.

9 Conclusions

A reference version of an Enterprise Wallet has been developed, featuring issuance, verification, and revocation capabilities that require integrations with the services of an Authentic Source for proper functioning. These integrations, particularly regarding data retrieval, allow the Wallet to be used as a service by external applications, although this process sacrifices form-like interfaces for data input.

This implementation is intended to serve as a foundational element for the construction of new Wallets with enhanced and expanded capabilities, as well as a starting point for the emergence of new discussions and debates about its overall evolution and the identification of improvement areas.

Additionally, with this initial version of the reference Wallet now available, various pilot scenarios in the project may choose to utilize it, particularly in cases where direct support from a technology partner is not available.

Regarding next steps, the reference implementation may be extended in future versions to include new features, such as expanded support for revocation mechanisms and the capability to manage accreditations. Likewise, as EBSI is an evolving project, the reference implementation will need to adapt to any changes within the framework.

Project co-funded by the European Union under the Digital Europe Programme. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union. Neither the European Union nor the granting authority can be held responsible for them.

10 References

[1] T. Lodderstedt, K. Yasuda e T. Looker, «OpenID for Verifiable Credential Issuance,» 30 December 2022. [Online]. Available: https://openid.net/specs/openid-4-verifiable-credential-issuance-1_0-10.html.

[2] O. Terbu, T. Lodderstedt, K. Yasuda, A. Lemmon e T. Looker, «OpenID for Verifiable Presentations,» 30 December 2022. [Online]. Available: https://openid.net/specs/openid-4-verifiable-presentations-1_0-14.html.

[3] W3C, «Verifiable Credentials Data Model v1.1,» 03 March 2022. [Online]. Available: https://www.w3.org/TR/vc-data-model/.

[4] W3C, «Decentralized Identifiers (DIDs) v1.0,» 19 July 2022. [Online]. Available: https://www.w3.org/TR/did-core/.

[5] EBSI, «DID Method for Legal Entities,» 26 July 2024. [Online]. Available: https://hub.ebsi.eu/vc-framework/did/legal-entities.

[6] W3C, «Verifiable Credentials Data Model v2.0,» 14 October 2024. [Online]. Available: https://www.w3.org/TR/vc-data-model-2.0/.

[7] DIF, «Presentation Exchange 2.0.0,» 03 November 2022. [Online]. Available: https://identity.foundation/presentation-exchange/spec/v2.0.0/.

[8] W3C, «Verifiable Credentials Status List v2021,» 27 April 2023. [Online]. Available: https://www.w3.org/TR/2023/WD-vc-status-list-20230427/.

[9] EBSI, «How to issue Verifiable Credentials,» 16 August 2024. [Online]. Available: https://hub.ebsi.eu/conformance/learn/verifiable-credential-issuance.

Project co-funded by the European Union under the Digital Europe Programme. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union. Neither the European Union nor the granting authority can be held responsible for them.

11 Resources

- Resource 1. **D3.6 Reference Implementation User Manual**: Provided as a separate document in the GitHub Repository of the Backend component, it shows, with examples how to interact with the administration interface to perform user management and the definition of issuance and verification flows.
- Resource 2. **Enterprise Wallet OpenAPI**: Provided as a separate document in the GitHub Repository of the Backend component, it contains the OpenAPI specification document of the Enterprise Wallet reference implementation.
- Resource 3: **Reference Implementation Repository**: Main repository with the code of the reference implementation. It is available at: https://github.com/izertis/identfy

Project co-funded by the European Union under the Digital Europe Programme. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union. Neither the European Union nor the granting authority can be held responsible for them.