VErifiable Credentials & Trusted Organisations Registries

ebsi

REFERENCE WALLET INTEGRATIONS MANUAL

INDICE

List of Figures

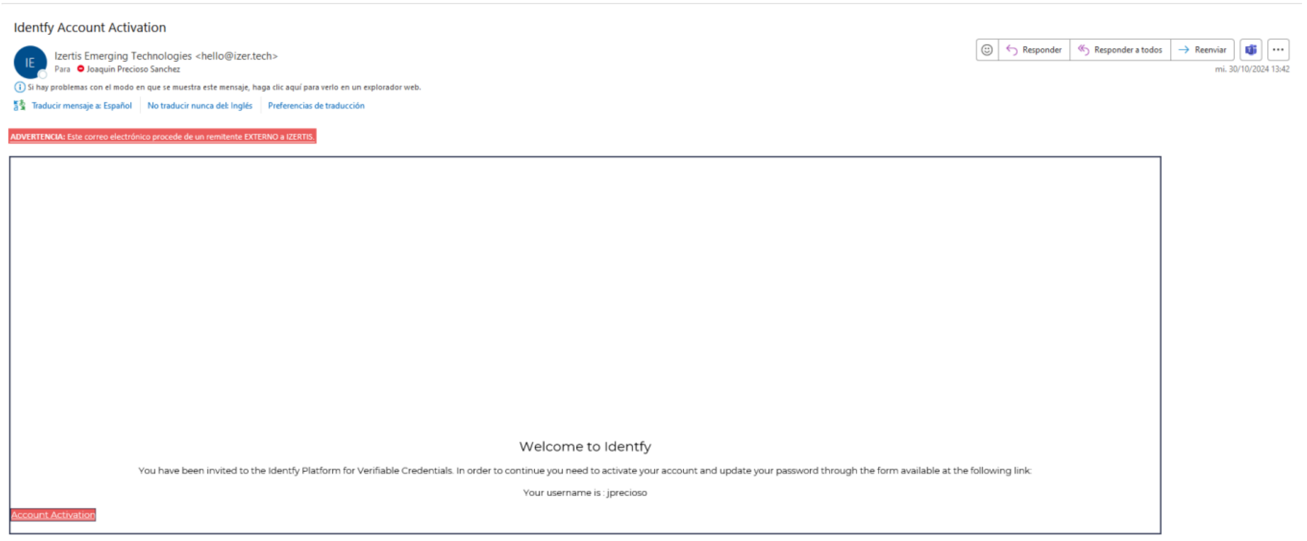
List of Tables

1 PURPOSE

The purpose of this document is to provide guidance enabling organizations to integrate the Enterprise Wallet component (Identfy) with their business systems, for the purpose of issuing, verifying, or revoking Verifiable Credentials.

1.1 Prerequisites and Considerations

The organization using the Enterprise Wallet (https://identfy.izer.tech) must have been onboarded in Identfy by the support team (IZERTIS). At least one user (previously provided by the organization) will have been granted access and received an informational email to activate their user account:



Identfy Account Activation

Izertis Emerging Technologies <hello@izer.tech>
Para Joaquin Precioso Sanchez

mi. 30/10/2024 13:42

Si hay problemas con el modo en que se muestra este mensaje, haga clic aquí para verlo en un explorador web.

Traducir mensaje a: Español | No traducir nunca del: Inglés | Preferencias de traducción

ADVERTENCIA: Este correo electrónico procede de un remitente EXTERNO a IZERTIS.

Welcome to Identfy

You have been invited to the Identfy Platform for Verifiable Credentials. In order to continue you need to activate your account and update your password through the form available at the following link:

Your username is : jprecioso

Account Activation

Illustration 1: Account Activation Email

Once the user has been activated, they will gain access to the Enterprise Wallet application (https://identfy.izer.tech), accessible via email and password.

The functionalities available will be limited by the role and permissions assigned to the user. In the context of integration testing covered by this document, it is important to understand at least the following concepts and resources provided by the Enterprise Wallet, which can be accessed from the left-hand menu (fig. 2)

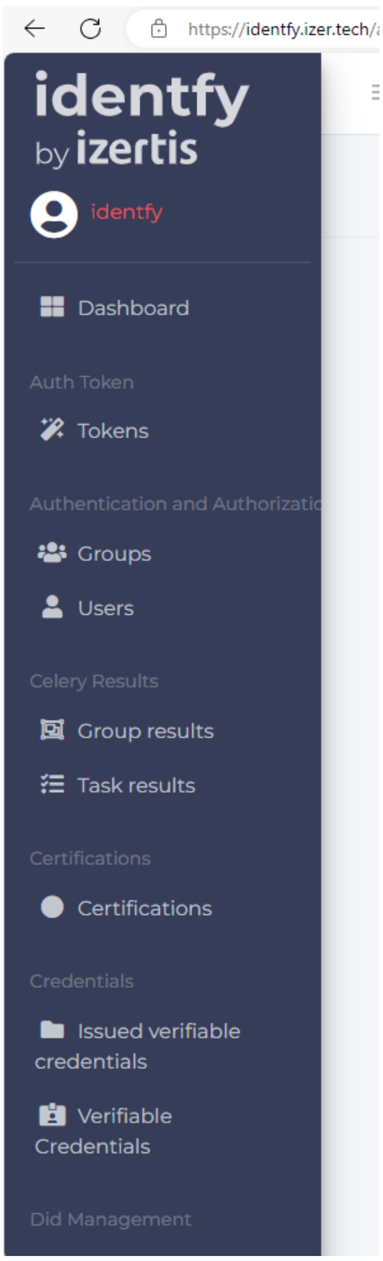


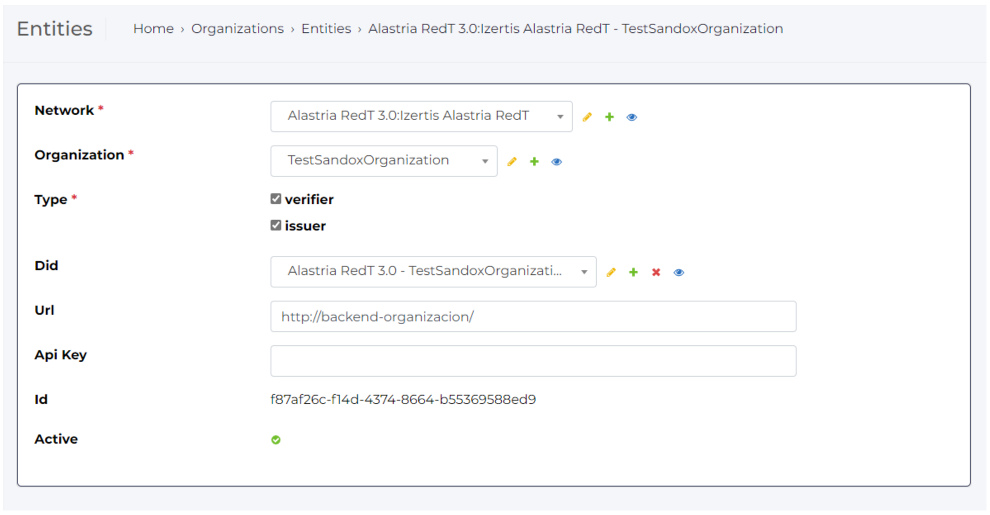
Illustration 2: Enterprise Wallet Functionalities

Entity

To enable an organization to operate on a network, an associated Entity must be defined. The Entity contains necessary information for integrating the Enterprise Wallet with the organization's business systems, defined as “Authentic Source”.

Relevant fields within the Entity include:

- **Id**: The unique identifier of the Entity in the Enterprise Wallet. It is used during credential issuance processes as part of the data exchanged between the Enterprise Wallet and the business systems.
- **URL**: This field allows registering the REST service that the Enterprise Wallet will use to communicate with the business systems during issuance, verification, or revocation processes (more details in section 3 Technical Explanation of the Integrations).
- **API Key**: If the organization protects the REST service using an API Key mechanism, this key is recorded in this field and will be sent by the Enterprise Wallet during every interaction with the organization's REST service.
- **DID**: The decentralized identifier used by the Entity. One is automatically generated when the organization is registered.
- **Type**: Indicates whether the organization will act as a credential issuer, verifier, or both.


Entities
Home › Organizations › Entities › Alastria RedT 3.0:Izertis Alastria RedT - TestSandoxOrganization

Network *	Alastria RedT 3.0:Izertis Alastria RedT
Organization *	TestSandoxOrganization
Type *	☑ verifier ☑ issuer
Did	Alastria RedT 3.0 - TestSandoxOrganizati...
Url	http://backend-organizacion/
Api Key	
Id	f87af26c-f14d-4374-8664-b55369588ed9
Active	✓

Illustration 3: Entities Screen - Enterprise Wallet

2 FUNCTIONAL EXPLANATION OF THE DIFFERENT TYPES OF INTEGRATIONS

2.1 Introduction

Conceptually, the Enterprise Wallet is the component that enables an organization to issue, verify, or revoke Verifiable Credentials based on data residing in the organization's internal systems (CRMs, ERPs, databases, etc.). It can be understood as a "bridge" between the user's wallet (Holder Wallet), which stores and presents credentials, and the business systems (hereafter referred to as *Authentic Source*) that contain the data and perform the related business logic.

The Enterprise Wallet does **not** require direct access to internal data repositories or applications (CRM, ERP, databases). Instead, it operates by consuming a set of REST services that must be provided by the organization. These services must comply with a specific OpenAPI/Swagger specification (further detailed in Section 5 – Annexes).

Finally, it is important to introduce the concept of an **Issuance Flow**, which refers to the process through which Verifiable Credentials are issued and managed within a digital identity ecosystem. To issue a credential, the subject's identity must first be determined. Depending on the mechanisms available within the organization to resolve this, three types of flows may be used:

- **In-Time Issuance Flow**
- **Pre-Authorized Issuance Flow**
- **Deferred Issuance Flow**

This section provides a detailed explanation of each type of flow, including their characteristics and relevant scenarios.

It is important to note that **it is not mandatory to implement all three flows.** An organization may choose the one that best suits its specific use cases and internal requirements.

3 TECHNICAL EXPLANATION OF THE INTEGRATIONS

3.1 Introduction

The Enterprise Wallet must communicate with the *Authentic Source* by consuming a set of REST services that the organization using the Enterprise Wallet must develop. These services must comply with the specifications indicated in the Annex section, ensuring availability and communication accessibility.

Depending on whether the organization takes the role of **credential issuer**, **credential verifier**, or **both**, the corresponding endpoints must be implemented.

Below is a summary of the required REST services:

Endpoint	**VERBO HTTP**	**Descripción**	**Rol**
/credentials/external-data	GET	Allows the Enterprise Wallet to retrieve the data to be used in the issuance of the Verifiable Credential	Credential Issuer
	POST	Notifies the Authentic Source that a credential has been successfully delivered to the user.	Credential Issuer
/presentations/external-data	POST	Sends the information obtained from the Verifiable Presentation to the backend of the organization	Credential Verifier

Table 1: Summary of endpoints to implement

4 CREDENTIAL ISSUANCE INTEGRATIONS

4.1.1 In-Time Issuance Flow

In the In-Time flow, the Verifiable Credential is issued to the user at the moment of request. The OpenID protocol is used for user authentication.

This flow is simple to implement but allows the user to be identified only through their DID, as no additional identifier is included during the process.

The Authentic Source must be capable of identifying the user and retrieving or generating the credential data using only the DID. If this is not possible, alternative flows described in this document should be considered.

Advantages	Disadvantages
Simple to implement. No need for an additional authentication system.	User can only be identified by DID.

Table 2: Comparison - In-Time Issuance Approach

4.1.2 Pre-Authorized Issuance Flow

In this flow, authorization/authentication is carried out through external processes using protocols that may differ from OpenID. As a result, there is no need to know the user's DID to identify them.

To better illustrate this flow, consider the following example and sequence of steps:

1. A user accesses a web application with its own authentication/authorization mechanism.
2. The user identifies themselves in the web application and requests the issuance of a Verifiable Credential.

3. The application accepts the request and generates a unique token, which it gives to the user. The user can then present this token to the Enterprise Wallet responsible for issuing the credential.
4. Upon receiving the token, the Enterprise Wallet sends it to the Authentic Source for validation.
5. If valid, the Authentic Source must respond with the credential data associated with the token, and thus, with the user for whom the credential is intended.

This flow requires the same REST integration as the In-Time flow; however, instead of identifying the user by DID, a token is used. The token must be managed by the organization within its systems.

Advantages	Disadvantages
Allows use of alternative authentication/authorization methods. Supports business logic-based authorization.	Authentic Source must implement its own auth mechanism. Token management logic must be implemented.

Table 3: Comparison - Pre-Authorized Issuance Approach

Note: The session token approach is the most secure to prevent re-entry attacks or duplicate requests (especially when combined with expiration mechanisms). However, this token may be replaced by another data element managed and known by the organization's business systems (e.g., a customer ID in a database). This option should be considered for testing only, not for production use cases.

4.1.3 Deferred Issuance Flow

The Deferred Issuance flow is similar to the In-Time flow, except that the credential is not issued immediately at the end of the process. Instead, the user is given an **acceptance code**, which can later be exchanged for the Verifiable Credential.

This flow allows for deferring credential issuance until a specific condition is met. For example:

- The credential data might not yet be available.
- Additional human validation of the request might be required.

In any case, when the Enterprise Wallet receives a request for this type of credential, it forwards it to the Authentic Source, which must respond with the acceptance code.

The Authentic Source must also provide a second endpoint that enables the exchange of this code for the credential data. Therefore, this flow requires **two integrations**: one for obtaining the acceptance code, and another for redeeming it.

It is worth noting that the user may attempt to redeem the code before the credential is available. The Authentic Source must be prepared to handle such cases. The simplest approach is to return either a new acceptance code or the same one.

Additionally, this flow can be **combined with the Pre-Authorized flow.** In that case, the user identifier submitted during the registration phase is the session token instead of the DID.

Advantages	**Disadvantages**
Enables deferred issuance based on business logic.	Requires more integrations. Authentic Source must manage acceptance token validity. Ideally, the Authentic Source should implement a notification mechanism to alert the Holder when the credential is available

Table 4: Comparison - Deferred Issuance Approach

5 CREDENTIAL VERIFICATION INTEGRATIONS

Credential verification requires an additional endpoint through which the Enterprise Wallet will deliver the data extracted from the credentials presented by the user via a Verifiable Presentation.

This presentation serves a dual purpose:

- To transmit the extracted data, which may be of interest to the Authentic Source.
- To request additional verification of that data in accordance with the organization's business logic.

Although the Enterprise Wallet is capable of verifying whether the content of a credential complies with a schema, it **cannot determine the factual correctness of the information.** For example, if a credential contains an identifier, only the Authentic Source can confirm whether that identifier actually exists or is valid.

For this reason, the integration endpoint must return a verified value as true or false, based on whether the request satisfies the organization's internal validation criteria.

The specification for this endpoint can be found in **Section 5.2 – Annexes.**

This is a POST method, where the Wallet sends:

- The DID of the user who presented the Verifiable Presentation.
- The extracted claims from the presented credentials.

Note that these claims represent only a **subset of the full credential data**, limited to what is explicitly requested in the presentation definition. For example, if the user is asked to present only a birth date, then only that data will be sent, **not the entire credential.** Therefore, it is essential to define presentation requests carefully.

In addition to the extracted claims, the Wallet also sends a **state** parameter. This is used when generating a QR code for a verification process and enables linking the verification result to an actual resource or access on the Authentic Source's side.

Finally, the field valid is a boolean that indicates whether the verification was successfully carried out. If set to false, **no credential data** will be sent. This attribute is mainly informative, but when used in combination with the state, it allows the Authentic Source to determine whether access to a resource should be denied—potentially triggering any necessary actions or safeguards.

In order to delve deeper into the contents of claimsData, an example will be presented. Consider the following ***PresentationDefinition***:

```
{
  "id": "<any id, random or static>",
  "format": {
    "jwt_vc": {
      "alg": [
        "ES256"
      ]
    },
    "jwt_vp": {
      "alg": [
        "ES256"
      ]
    }
  },
  "input_descriptors": [
    {
      "id": "example-vp",
      "format": {
        "jwt_vc": {
          "alg": [
            "ES256"
          ]
        }
      },
      "constraints": {
        "fields": [
          {
            "id": "credentialType",
            "path": [
              "$.vc.type"
            ],
            "filter": {
              "type": "array",
              "contains": {
                "const": "EducationDiploma"
              }
            }
          },
          {
            "id": "random_data",
            "path": [
              "$.vc.credentialSubject.randomData"
            ],
            "filter": {
              "type": "string"
            }
          }
        ]
      }
    }
  ]
}
```

An example of the data that could be received is as follows:

```
{
  "valid": true,
  "holderDid":
"did:key:z2dmzD81cgPx8Vki7JbuuMmFYrWPgYoytykUZ3eyqht1j9KbrzTG7NZYJAJvfMbSDmMeWtKNeWA
TywxfBBHwbRsYn6o3VV6AeGih9jpQE8uv3RMQhVzEET5T4uKmXjxhWFRYHDHwXW2bPMVUuT4bDcrAHYSf7DG
vNzUWRBzd3AH8Cwcq2C",
  "claimsData": {
    "example-vp": {
      "credentialType": [
        "VerifiableAttestation",
        "VerifiableCredential",
        "EducationDiploma"
      ],
      "random_data": "aaabbbccc"
    }
  },
  "state": "9f8ae33e-7361-4fe1-bb6d-0147aa9bc3db"
}
```

Note how the identifiers present in “claimsData” are the same as those used in the definition.

6 ANNEXES

6.1 Endpoint Specification: /credentials/external-data

The specification for this endpoint is as follows:

```yaml
openapi: 3.0.3
info:
  title: Credential Data Integrations
  description: |-
    Endpoint specification for Credential Data integrations
  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
          in: query
          schema:
            type: string
          required: false
      responses:
        '200':
          description: Successful response with VC Data
          content:
            application/json:
              schema:
                type: object
                additionalProperties: true


    post:
      description: Notify about a verifiable credential that has been
successfully issued.
```

```yaml
      requestBody:
        content:
          application/json:
            schema:
              type: object
              properties:
                vc_type:
                  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
                  format: date-time
                  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
                  format: date-time
                  description: The issuance date of the verifiable credential
              required:
                - vc_type
                - did
                - vc_id
                - nbf
                - pre_code
                - issuance_date
      responses:
        '200':
          description: Successful response
          content:
            application/json:
              schema:
                type: object
                properties:
                  message:
                    type: string
                    description: Confirmation message for the successful
notification
```

Descriptive Summary of Parameters:

Parameter	Description
VC_TYPE	The identifier of the credential type to be issued. Defined in the Enterprise Wallet under "Issuance Flows."
USER_ID	The code used to identify the requester of the Verifiable Credential. It may be a user token (as explained in section 2.2) or another internal identifier used by the organization.

Table 5: Endpoint Parameters

6.2 Endpoint Specification: /presentations/external-data:

The specification for this endpoint is as follows:

```
/presentations/external-data:
    post:
      description: Notify about a verifiable credential that has been succcesfull
isued.
      requestBody:
        content:
          application/json:
            schema:
              type: object
              properties:
                valid:
                  type: boolean
                  description: Flag that indicate if the verification was
succcesfull or not.
                hoderDid:
                  type: string
                  description: The user DID.
                state:
                  type: string
                  description: The state sended 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
          content:
            application/json:
              schema:
                type: object
                properties:
                  verified:
                    title: Verified
                    type: boolean
                required:
                  - verified
```