Table of Contents

HYPER-AI T6.1 Lead by ODINS

hyper-ai

Issue report form

Please, before reporting your issue, make sure you have reviewed the following sections of this page, as they may contain information about it and potentially how to solve, or the ongoing process in solving it.

Afterwards, if your issue has not been addressed in the following sections, please copy the following issue report form into an email, fill it out and send it to jsanchez@odins.es or fjromero@odins.es.

1. Email subject: `[HYPER-AI] Issue Report - <topic>`
2. Reported by:
3. Email:
4. Tasks involved/Partners involved:
5. Broker Entities in particular involved:
6. Severity (select from Minor, Moderate, Major and Critical):
7. Summary:
8. Description:
9. Error messages obtained:
10. Requests that produce the errors (so that we can reproduce the issue):
11. Time interval (UTC time) during which these errors have occurred:

We will try to resolve your issue as soon as possible. Thank you very much and sorry for the inconvenience.

Introduction to DIDs

DID (Decentralized Identifiers) are a type of unique and decentralized identifier that allows identity verification without the need for a centralized authority. These identifiers are designed to provide sovereign control over identity, meaning that the owner of a DID can manage their identity without relying on intermediaries such as governments, companies, or organizations.

DIDs enable authentication, digital signatures, and credential verification in a secure and decentralized manner. They are generally associated with public key cryptography to validate the authenticity of interactions and documents.

```
{
    "@context": "https://www.w3.org/ns/did/v1",
    "assertionMethod": [
        "did:fabric:z218q5McPufPy8NqCG29DRNq#bee3b922c76946f9b14cf0c730c00e99"
    ],
    "authentication": [
        "did:fabric:z218q5McPufPy8NqCG29DRNq#bee3b922c76946f9b14cf0c730c00e99"
    ],
    "id": "did:fabric:z218q5McPufPy8NqCG29DRNq",
    "verificationMethod": [
        {
            "controller": "did:fabric:z218q5McPufPy8NqCG29DRNq",
            "id": "did:fabric:z218q5McPufPy8NqCG29DRNq#bee3b922c76946f9b14cf0c730c00e99",
            "publicKeyJwk": {
                "alg": "ES256K",
                "crv": "secp256k1",
                "kid": "bee3b922c76946f9b14cf0c730c00e99",
                "kty": "EC",
                "use": "sig",
                "x": "kg-qcoe5Sg8uqqCyzDDGOFa5XPx6jHAFN-U1Xo__NUE",
                "y": "EZ-Jg8lQVERffibrQU45Keumxh912jrKIePVGK8GnDc"
            },
            "type": "EcdsaSecp256k1VerificationKey2019"
        }
    ]
}
```


Taking the example shown in the image above.

The **@context** field defines the context in which the DID will be interpreted. In this case, it uses the W3C standard for DIDs: “https://www.w3.org/ns/did/v1”. It ensures that the properties of the DID document are correctly interpreted.

The **id** field is the decentralized identifier of the DID. In this case: “did:fabric:z218q5McPufPy8NqCG29DRNq”. The prefix “did:fabric:” suggests that this DID is based on a Hyperledger Fabric network, which is a blockchain technology.

The **assertionMethod** field lists digital signature methods that can be used to assert verifiable claims (e.g., sign digital credentials). It refers to a specific verification key using a fragment (#bee3b922c76946f9b14cf0c730c00e99).

The **authentication** field defines valid authentication methods for this DID. Authentication allows the DID owner to prove their identity through the use of cryptographic keys. In this case, it uses the same key as in assertionMethod.

The **verificationMethod** field contains a list of identity verification methods. In this case, it includes a public key in JWK (JSON Web Key) format used for signing and verifying digital signatures.

Within **verificationMethod**, the controller field identifies who controls this key. In this case, it is the same DID (“did:fabric:z218q5McPufPy8NqCG29DRNq”).

The id field represents the unique identifier of this verification key.

The **publicKeyJwk** field represents the public key in JWK format. Within this field:

“alg”: “ES256K” indicates that the cryptographic algorithm is ECDSA with SHA-256 on secp256k1 (used in Bitcoin and Ethereum).

“crv”: “secp256k1” indicates the elliptic curve used.

“kid”: “bee3b922c76946f9b14cf0c730c00e99” is the unique identifier of the key.

“kty”: “EC” indicates that the key type is Elliptic (EC).

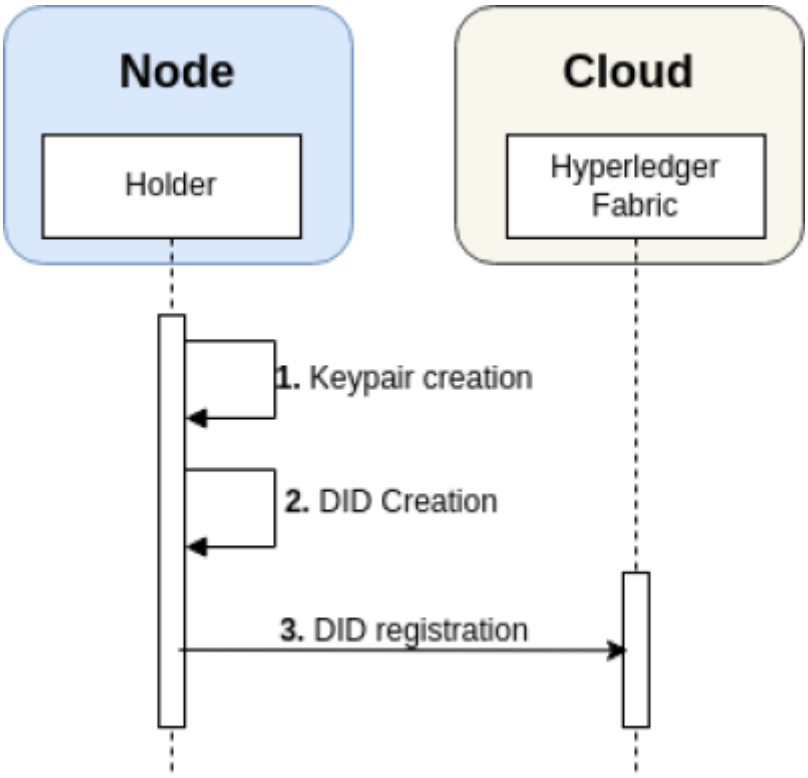
“use”: “sig” indicates that the key is used for digital signing.

“x” and “y” represent the coordinates of the public key on the elliptic curve.

The **type** field specifies the type of key, in this case, “EcdsaSecp256k1VerificationKey2019”, which is an ECDSA key on the secp256k1 curve.

Operations with DIDs

The first step after deploying all security components is to create a DID to uniquely identify each device and register it in the Blockchain. This process is carried out by the Holder of each device. This procedure defines the interaction of the different components for the creation and registration of the DID of a Node.



1. A key pair is generated using the ES256K algorithm. Both the public and private keys are stored locally within the Holder, specifically in the ‘data/key’ folder of each pod's Holder. This folder, along with all directories under ‘/data,’ is created dynamically at runtime.
2. Once the key pair is created, the Holder of each pod generates a DID. The DID consists of a DID

Id and a DID Document. The DID Document includes relevant information such as the identifier and the 'verificationMethod,' which provides the necessary details to retrieve the public key associated with that DID.

3. The DID is then stored both locally, in the 'data/did/created' folder of each pod's Holder, and in the Hyperledger Fabric (Blockchain). Within the blockchain, the DID is saved in a 'key-value' format, where the key represents the DID Id and the value corresponds to the DID Document.

Use of DIDs

To register a DID on the Blockchain, a POST request must be sent to the Blockchain API. This request should include the DID to be registered in the 'Did' header and the corresponding DID Document in the request body. If the registration is successful, the API will respond with a '200 OK' status.

```
fjromero curl -X POST "https://heimdall.odins.es/fabric/did/register" \
  -H "Did: did:fabric:z218q5McPufPy8NqCG29DRNq" \
  -H "Content-Type: application/json" \
  -d '{
    "@context": "https://www.w3.org/ns/did/v1",
    "assertionMethod": [
        "did:fabric:z218q5McPufPy8NqCG29DRNq#bee3b922c76946f9b14cf0c730c00e99"
    ],
    "authentication": [
        "did:fabric:z218q5McPufPy8NqCG29DRNq#bee3b922c76946f9b14cf0c730c00e99"
    ],
    "id": "did:fabric:z218q5McPufPy8NqCG29DRNq",
    "verificationMethod": [
        {
            "controller": "did:fabric:z218q5McPufPy8NqCG29DRNq",
            "id": "did:fabric:z218q5McPufPy8NqCG29DRNq#bee3b922c76946f9b14cf0c730c00e99",
            "publicKeyJwk": {
                "alg": "ES256K",
                "crv": "secp256k1",
                "kid": "bee3b922c76946f9b14cf0c730c00e99",
                "kty": "EC",
                "use": "sig",
                "x": "kg-qcoe5Sg8uqqCyzDDGOFa5XPx6jHAFN-U1Xo__NUE",
                "y": "EZ-Jg8lQVERffibrQU45Keumxh912jrKIePVGK8GnDc"
            },
            "type": "EcdsaSecp256k1VerificationKey2019"
        }
    ]
  }'
DID 'did:fabric:z218q5McPufPy8NqCG29DRNq' successfully registered in Hyperledger Fabric%
```

To retrieve a DID Document from the Blockchain, a GET request should be made to the Blockchain API, specifying the desired DID in the 'Did' header. If the DID is found in the Blockchain, the response will include a '200 OK' status along with the DID Document in the response body.

```
fjromero  curl -X GET "https://heimdall.odins.es/fabric/did/obtain" \
     -H "did: did:fabric:z218q5McPufPy8NqCG29DRNq"
{
    "@context": "https://www.w3.org/ns/did/v1",
    "assertionMethod": [
        "did:fabric:z218q5McPufPy8NqCG29DRNq#bee3b922c76946f9b14cf0c730c00e99"
    ],
    "authentication": [
        "did:fabric:z218q5McPufPy8NqCG29DRNq#bee3b922c76946f9b14cf0c730c00e99"
    ],
    "id": "did:fabric:z218q5McPufPy8NqCG29DRNq",
    "verificationMethod": [
        {
            "controller": "did:fabric:z218q5McPufPy8NqCG29DRNq",
            "id": "did:fabric:z218q5McPufPy8NqCG29DRNq#bee3b922c76946f9b14cf0c730c00e99",
            "publicKeyJwk": {
                "alg": "ES256K",
                "crv": "secp256k1",
                "kid": "bee3b922c76946f9b14cf0c730c00e99",
                "kty": "EC",
                "use": "sig",
                "x": "kg-qcoe5Sg8uqqCyzDDGOFa5XPx6jHAFN-U1Xo__NUE",
                "y": "EZ-Jg8lQVERffibrQU45Keumxh912jrKIePVGK8GnDc"
            },
            "type": "EcdsaSecp256k1VerificationKey2019"
        }
    ]
}%
```

Endpoints

Folder postman contains a JSON file that can be exported to POSTMAN to test all endpoints described below:

0. CI/CD

0.1 Decentralized Identifiers

- **Endpoint**: /test/did

- **Method**: GET
- **Description**: Returns the name of the channel where the DID Smart Contract is deployed. Useful to check that the Smart Contract has been successfully deployed.

```
fjromero  curl -X GET "https://heimdall.odins.es/test/did"
hyperai%
```

1. Decentralized Identifiers

1.1 Register DID in Blockchain

- **Endpoint**: /fabric/did/register

- **Method**: POST
- **Header**:
    - **Content-Type**: application/json
    - **did**: <DID Id> (where 'DID Id' is the DID id to be registered in Blockchain)
- **Body**: <DID Document> (where 'DID Document' is the JSON of the DID Document you want to register in Blockchain)
- **Description**: Register a DID Document associated with a DID Id in Blockchain.

1.2 Get DID from Blockchain

- **Endpoint**: /fabric/did/obtain

- **Method**: GET
- **Header**:
    - **did**: <DID Id> (DID Id to be retrieved from Blockchain)
- **Description**: Retrieve the DID Document associated with the DID id indicated in the 'did' header from Blockchain.