5

Operative Specification

An operational specification provides a comprehensive overview of how the software is expected to function in various operating conditions. It serves as a road map for software development and testing and ensures that the final product meets the user's requirements and expectations.

5.1 Libraries

The list of the different libraries used to integrate into the i3-MARKET framework is shown below.

Auditable accounting library:

- The auditable accounting component is a service that includes an API to automate the process of logging and auditing interactions between components and record the registries in the blockchain. The API of the auditable accounting is accessed through the Backplane API gateway. Additionally, the auditable accounting component can be accessed directly from any internal component of the platform.
- License: MIT.
- Source code: https://gitlab.com/i3-market-v3-public-repository/sp3-scgbssw-aa-auditableaccounting.
- Prerequisites: Node.js, Docker, and Docker Compose.

Wallet client library:

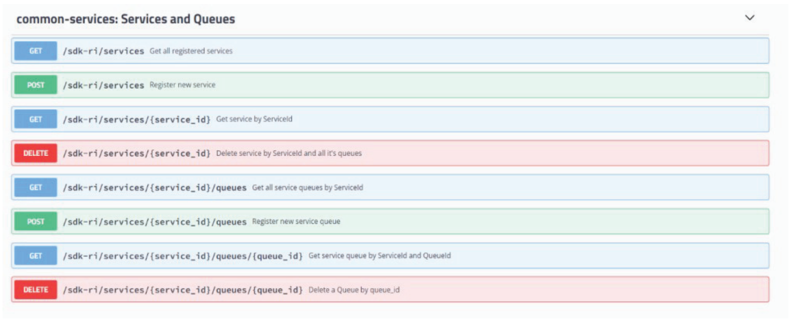
- This package defines how to interact with wallets by means of a typescript interface. Furthermore, it provides a default implementation called BaseWallet. It uses an interface called KeyWallet to delegate the complexity of key management to other packages like SW Wallet. Both interfaces are listed below.
- License: Apache License 2.0.

- Source code: https://gitlab.com/i3-market-v3-public-repository/sp3-scgbssw-i3mwalletmonorepo.
- Prerequisites: Node.js.

5.2 i3-MARKET APIs

The update compared to R1 in terms of common services is the following:

i) Notification manager common services: The functionalities related with notification services and queues were the scope of R2 and R3 and are listed in Figure 5.1.

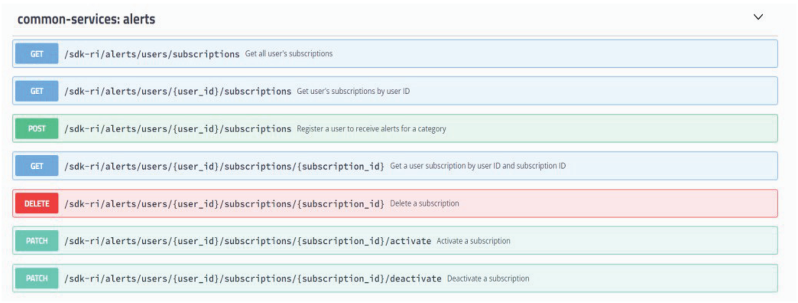


common-services: Services and Queues

GET	/sdk-ri/services	Get all registered services
POST	/sdk-ri/services	Register new service
GET	/sdk-ri/services/{service_id}	Get service by ServiceId
DELETE	/sdk-ri/services/{service_id}	Delete service by ServiceId and all it's queues
GET	/sdk-ri/services/{service_id}/queues	Get all service queues by ServiceId
POST	/sdk-ri/services/{service_id}/queues	Register new service queue
GET	/sdk-ri/services/{service_id}/queues/{queue_id}	Get service queue by ServiceId and QueueId
DELETE	/sdk-ri/services/{service_id}/queues/{queue_id}	Delete a Queue by queue_id

Figure 5.1 Services and queues common services.

ii) Alerts common services: The functionalities related with alerts were the scope of R2 and R3 and are listed in Figures 5.2, 5.3, and 5.4.

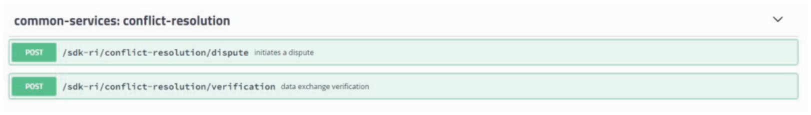


common-services: alerts

GET	/sdk-ri/alerts/users/subscriptions	Get all user's subscriptions
GET	/sdk-ri/alerts/users/{user_id}/subscriptions	Get user's subscriptions by user ID
POST	/sdk-ri/alerts/users/{user_id}/subscriptions	Register a user to receive alerts for a category
GET	/sdk-ri/alerts/users/{user_id}/subscriptions/{subscription_id}	Get a user subscription by user ID and subscription ID
DELETE	/sdk-ri/alerts/users/{user_id}/subscriptions/{subscription_id}	Delete a subscription
PATCH	/sdk-ri/alerts/users/{user_id}/subscriptions/{subscription_id}/activate	Activate a subscription
PATCH	/sdk-ri/alerts/users/{user_id}/subscriptions/{subscription_id}/deactivate	Deactivate a subscription

Figure 5.2 Alerts common services.

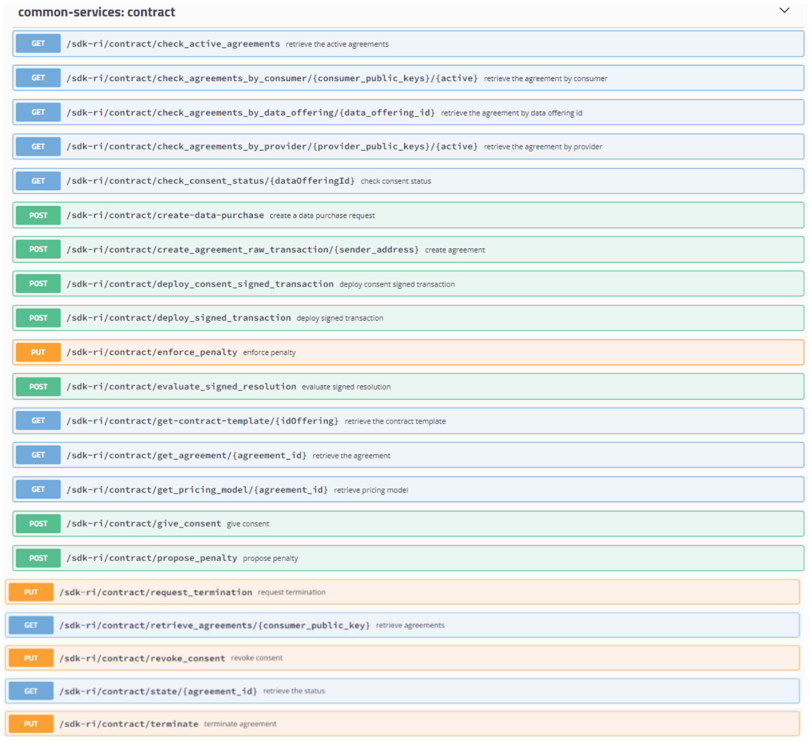
iii) Conflict resolution common services:
The functionalities related with contradictory conditions enabled by two methods as shown in Figure 5.3


common-services: conflict-resolution

POST /sdk-ri/conflict-resolution/dispute initiates a dispute

POST /sdk-ri/conflict-resolution/verification data exchange verification

Figure 5.3 Conflict resolution common services.


common-services: contract

GET /sdk-ri/contract/check_active_agreements retrieve the active agreements

GET /sdk-ri/contract/check_agreements_by_consumer/{consumer_public_keys}/{active} retrieve the agreement by consumer

GET /sdk-ri/contract/check_agreements_by_data_offering/{data_offering_id} retrieve the agreement by data offering id

GET /sdk-ri/contract/check_agreements_by_provider/{provider_public_keys}/{active} retrieve the agreement by provider

GET /sdk-ri/contract/check_consent_status/{dataOfferingId} check consent status

POST /sdk-ri/contract/create-data-purchase create a data purchase request

POST /sdk-ri/contract/create_agreement_raw_transaction/{sender_address} create agreement

POST /sdk-ri/contract/deploy_consent_signed_transaction deploy consent signed transaction

POST /sdk-ri/contract/deploy_signed_transaction deploy signed transaction

PUT /sdk-ri/contract/enforce_penalty enforce penalty

POST /sdk-ri/contract/evaluate_signed_resolution evaluate signed resolution

GET /sdk-ri/contract/get-contract-template/{idOffering} retrieve the contract template

GET /sdk-ri/contract/get_agreement/{agreement_id} retrieve the agreement

GET /sdk-ri/contract/get_pricing_model/{agreement_id} retrieve pricing model

POST /sdk-ri/contract/give_consent give consent

POST /sdk-ri/contract/propose_penalty propose penalty

PUT /sdk-ri/contract/request_termination request termination

GET /sdk-ri/contract/retrieve_agreements/{consumer_public_key} retrieve agreements

PUT /sdk-ri/contract/revoke_consent revoke consent

GET /sdk-ri/contract/state/{agreement_id} retrieve the status

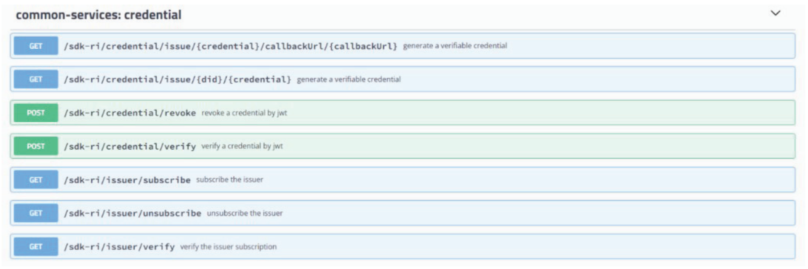
PUT /sdk-ri/contract/terminate terminate agreement

Figure 5.4 Contracts common services.

iv) Contracts common services: The functionalities related with smart contracts management were the scope of R2 and R3 and are listed in Figure 5.5.

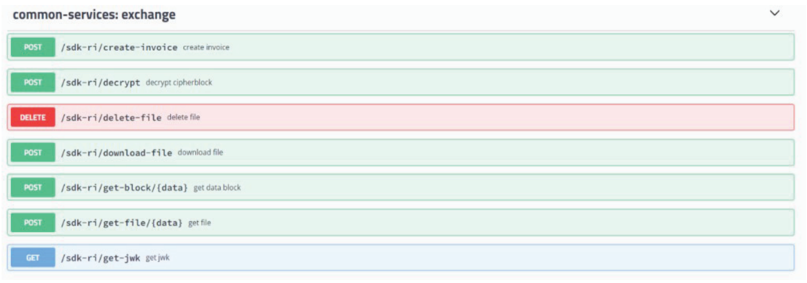
v) Credential common services: The functionalities related with authentication, identities, and credentials were the scope of R2 and R3 and are listed in 5.5.

vi) Exchange common services: The functionalities related with data exchange were the scope of R2 and R3 and are listed in Figure 5.6.



```
common-services: credential
GET    /sdk-ri/credential/issue/{credential}/callbackUrl/{callbackUrl}  generate a verifiable credential
GET    /sdk-ri/credential/issue/{did}/{credential}  generate a verifiable credential
POST   /sdk-ri/credential/revoke  revoke a credential by jwt
POST   /sdk-ri/credential/verify  verify a credential by jwt
GET    /sdk-ri/issuer/subscribe  subscribe the issuer
GET    /sdk-ri/issuer/unsubscribe  unsubscribe the issuer
GET    /sdk-ri/issuer/verify  verify the issuer subscription
```

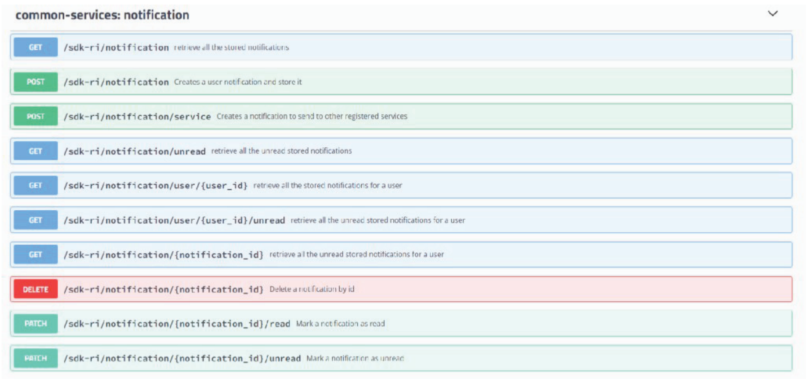
Figure 5.5 Contracts common services.



```
common-services: exchange
POST   /sdk-ri/create-invoice  create invoice
POST   /sdk-ri/decrypt  decrypt cipherblock
DELETE /sdk-ri/delete-file  delete file
POST   /sdk-ri/download-file  download file
POST   /sdk-ri/get-block/{data}  get data block
POST   /sdk-ri/get-file/{data}  get file
GET    /sdk-ri/get-jwk  get jwk
```

Figure 5.6 Exchange common services.

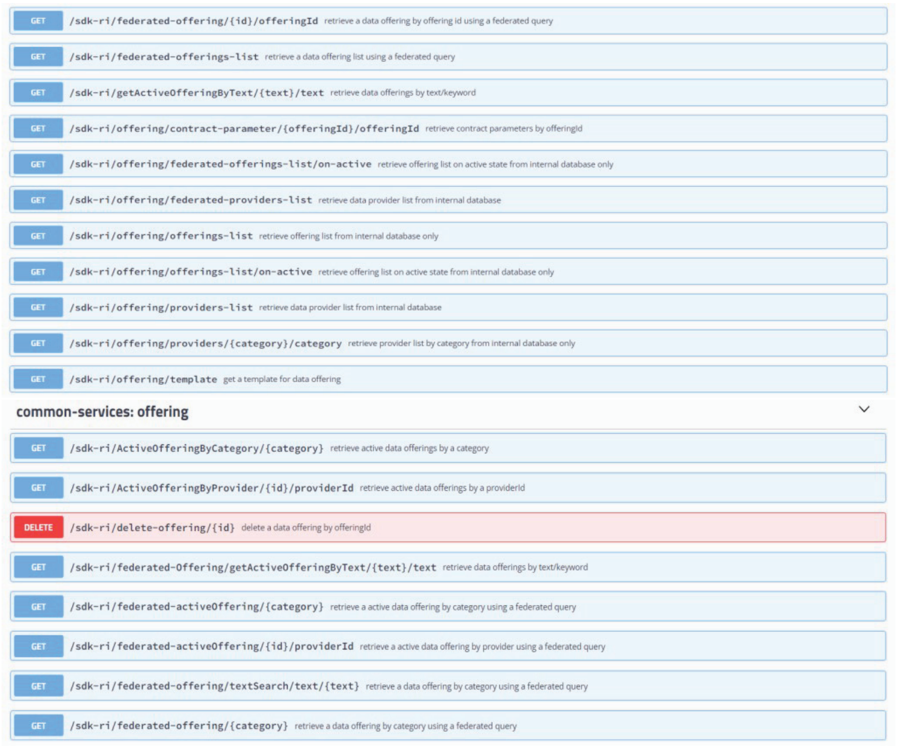
vii) Notification common services: The functionalities related with notifications were the scope of R2 and R3 and are listed in Figure 5.7.



```
common-services: notification
GET    /sdk-ri/notification  retrieve all the stored notifications
POST   /sdk-ri/notification  Creates a user notification and store it
POST   /sdk-ri/notification/service  Creates a notification to send to other registered services
GET    /sdk-ri/notification/unread  retrieve all the unread stored notifications
GET    /sdk-ri/notification/user/{user_id}  retrieve all the stored notifications for a user
GET    /sdk-ri/notification/user/{user_id}/unread  retrieve all the unread stored notifications for a user
GET    /sdk-ri/notification/{notification_id}  retrieve all the unread stored notifications for a user
DELETE /sdk-ri/notification/{notification_id}  Delete a notification by id
PATCH  /sdk-ri/notification/{notification_id}/read  Mark a notification as read
PATCH  /sdk-ri/notification/{notification_id}/unread  Mark a notification as unread
```

Figure 5.7 Notification common services.

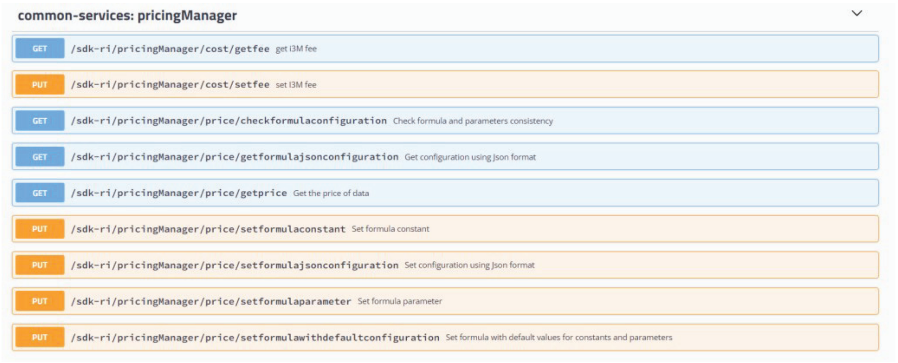
viii) Offering management common services: The functionalities related with data offering management were the scope of R2 and R3 and are listed in Figure 5.8.



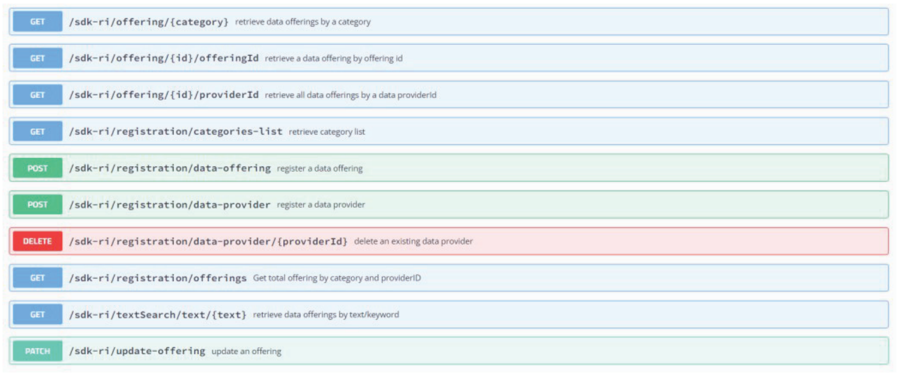
```
GET     /sdk-ri/federated-offering/{id}/offeringId  retrieve a data offering by offering id using a federated query
GET     /sdk-ri/federated-offerings-list  retrieve a data offering list using a federated query
GET     /sdk-ri/getActiveOfferingByText/{text}/text  retrieve data offerings by text/keyword
GET     /sdk-ri/offering/contract-parameter/{offeringId}/offeringId  retrieve contract parameters by offeringId
GET     /sdk-ri/offering/federated-offerings-list/on-active  retrieve offering list on active state from internal database only
GET     /sdk-ri/offering/federated-providers-list  retrieve data provider list from internal database
GET     /sdk-ri/offering/offerings-list  retrieve offering list from internal database only
GET     /sdk-ri/offering/offerings-list/on-active  retrieve offering list on active state from internal database only
GET     /sdk-ri/offering/providers-list  retrieve data provider list from internal database
GET     /sdk-ri/offering/providers/{category}/category  retrieve provider list by category from internal database only
GET     /sdk-ri/offering/template  get a template for data offering

common-services: offering

GET     /sdk-ri/ActiveOfferingByCategory/{category}  retrieve active data offerings by a category
GET     /sdk-ri/ActiveOfferingByProvider/{id}/providerId  retrieve active data offerings by a providerId
DELETE  /sdk-ri/delete-offering/{id}  delete a data offering by offeringId
GET     /sdk-ri/federated-Offering/getActiveOfferingByText/{text}/text  retrieve data offerings by text/keyword
GET     /sdk-ri/federated-activeOffering/{category}  retrieve a active data offering by category using a federated query
GET     /sdk-ri/federated-activeOffering/{id}/providerId  retrieve a active data offering by provider using a federated query
GET     /sdk-ri/federated-offering/textSearch/text/{text}  retrieve a data offering by category using a federated query
GET     /sdk-ri/federated-offering/{category}  retrieve a data offering by category using a federated query
```

Figure 5.8 Offering common services.

ix) Pricing managing common services: The functionalities related with pricing managing were the scope of R2 and R3 and are listed in Figure 5.9.



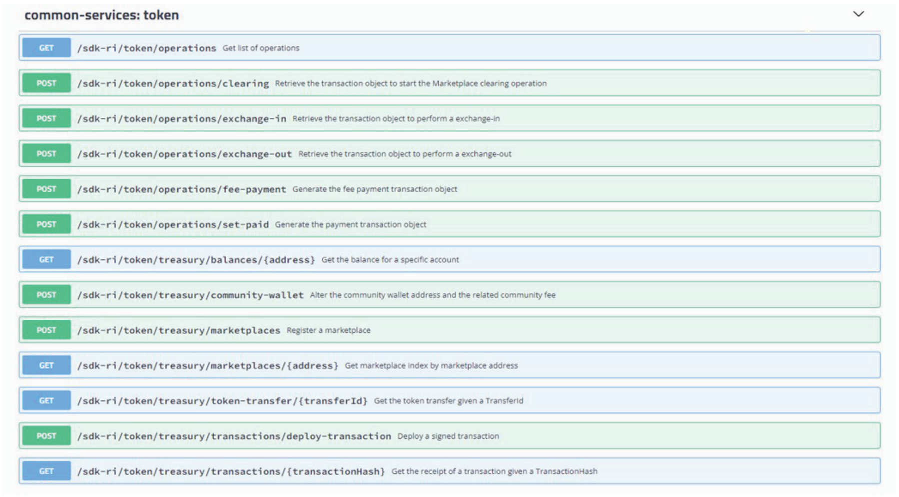
```
common-services: pricingManager

GET     /sdk-ri/pricingManager/cost/getfee  get i3M fee
PUT     /sdk-ri/pricingManager/cost/setfee  set I3M fee
GET     /sdk-ri/pricingManager/price/checkformulaconfiguration  Check formula and parameters consistency
GET     /sdk-ri/pricingManager/price/getformulajsonconfiguration  Get configuration using Json format
GET     /sdk-ri/pricingManager/price/getprice  Get the price of data
PUT     /sdk-ri/pricingManager/price/setformulaconstant  Set formula constant
PUT     /sdk-ri/pricingManager/price/setformulajsonconfiguration  Set configuration using Json format
PUT     /sdk-ri/pricingManager/price/setformulaparameter  Set formula parameter
PUT     /sdk-ri/pricingManager/price/setformulawithdefaultconfiguration  Set formula with default values for constants and parameters
```



```
GET     /sdk-ri/offering/{category}  retrieve data offerings by a category
GET     /sdk-ri/offering/{id}/offeringId  retrieve a data offering by offering id
GET     /sdk-ri/offering/{id}/providerId  retrieve all data offerings by a data providerId
GET     /sdk-ri/registration/categories-list  retrieve category list
POST    /sdk-ri/registration/data-offering  register a data offering
POST    /sdk-ri/registration/data-provider  register a data provider
DELETE  /sdk-ri/registration/data-provider/{providerId}  delete an existing data provider
GET     /sdk-ri/registration/offerings  Get total offering by category and providerID
GET     /sdk-ri/textSearch/text/{text}  retrieve data offerings by text/keyword
PATCH   /sdk-ri/update-offering  update an offering
```

Figure 5.9 Pricing common services.

x) Token managing common services: The functionalities related with token management were the scope of R2 and R3 and are listed in Figure 5.10.



```
common-services: token

GET   /sdk-ri/token/operations  Get list of operations
POST  /sdk-ri/token/operations/clearing  Retrieve the transaction object to start the Marketplace clearing operation
POST  /sdk-ri/token/operations/exchange-in  Retrieve the transaction object to perform a exchange-in
POST  /sdk-ri/token/operations/exchange-out  Retrieve the transaction object to perform a exchange-out
POST  /sdk-ri/token/operations/fee-payment  Generate the fee payment transaction object
POST  /sdk-ri/token/operations/set-paid  Generate the payment transaction object
GET   /sdk-ri/token/treasury/balances/{address}  Get the balance for a specific account
POST  /sdk-ri/token/treasury/community-wallet  Alter the community wallet address and the related community fee
POST  /sdk-ri/token/treasury/marketplaces  Register a marketplace
GET   /sdk-ri/token/treasury/marketplaces/{address}  Get marketplace index by marketplace address
GET   /sdk-ri/token/treasury/token-transfer/{transferId}  Get the token transfer given a TransferId
POST  /sdk-ri/token/treasury/transactions/deploy-transaction  Deploy a signed transaction
GET   /sdk-ri/token/treasury/transactions/{transactionHash}  Get the receipt of a transaction given a TransactionHash
```

Figure 5.10 Tokens common services.

5.3 SDKs

The layered SDK approach defined in the mechanism allows to adapt and extend existing data marketplaces to interface with the i3-MARKET Backplane.

Specifically, the layers that are part of the proposed solution for the SDK are the following:

- **SDK-core:** This layer aims to simplify the i3-MARKET SDK building process by generating client stubs for any i3-MARKET backend endpoint/API, defined with the OpenAPI (formerly known as Swagger) specification. In this way, therefore, the development team can better focus on the implementation and adoption of these backend endpoints or APIs.
- **SDK reference implementation (SDK-RI):** This layer aims to identify and provide a set of common services to be implemented for consuming available Backplane functionalities.
- **SDK-execution patterns (SDK-EP):** It is including the atomic functions that make use of Backplane API (via SDK) adding some business logic.
- **SDK Web-RI:** It is supporting the frontend or GUI integrating the common services provided by the SDK-RI and that can be reused and customized as part of the pilot specification and implementation defined in the context of WP5.

5.4 User Interfaces

To contextualize the i3-MARKET frontend or SDK Web-RI, it is important to introduce the SDK global approach and is shown in Figure 5.11. SDK Web-RI would be the top layer on the layered approach defined as part of the SDK solution for i3-MARKET.

i3-MARKET Web-RI provides a graphical user interface component, designed to use the reference implementation (SDK-RI) through a user interface to validate i3-MARKET functionalities from the user's point of view. It will be provided as an open-source component for the i3-MARKET implementation and for future pilots.

Web-RI can be used also by other market players to easily integrate with i3-MARKET and even set up a marketplace. Web-RI implements the following basic workflows:

- Register new data offerings and delete data offerings
- Search for offerings
- Create and sign smart contracts
- Purchase data
- Pay for data

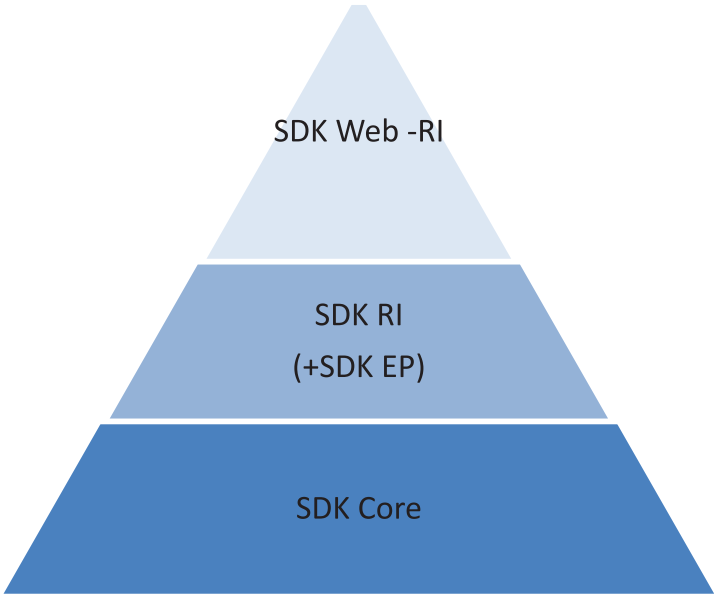


Figure 5.11 Implementation pyramid.

- Transfer data
- Rate data providers

This section aims to explain how an end-user can operate within the i3-MARKET user interface.

5.5 Install i3M Wallet

Go to repo URL (https://github.com/i3-Market-V3-Public-Repository/SP3-SCGBSSW-I3mWalletMonorepo/releases) and download the v2.5.6 version suitable for your operating system and do the following actions for:

- Windows operating system:
  - Download and execute wallet-desktop-v2.5.6-x64.exe.
  - The application is a standalone RAR file. Extract it and execute the i3M Wallet.exe file.
- MacOS operating system:
  - Open the dmg file and install the wallet desktop application.

- Linux operating system:
  - For Debian-based systems, you can use the deb package:
- # change x.x.x for the version.
- sudo dpkg-i wallet-desktop-x.x.x-amd64.deb.

5.6 Create a Wallet and a Consumer and/or Provider Identity in the Wallet

The first time a user initiates the application, a dialog asking for a password appears (see following pictures for more details). The user will have to introduce this password each time the application starts – see Figure 5.12.

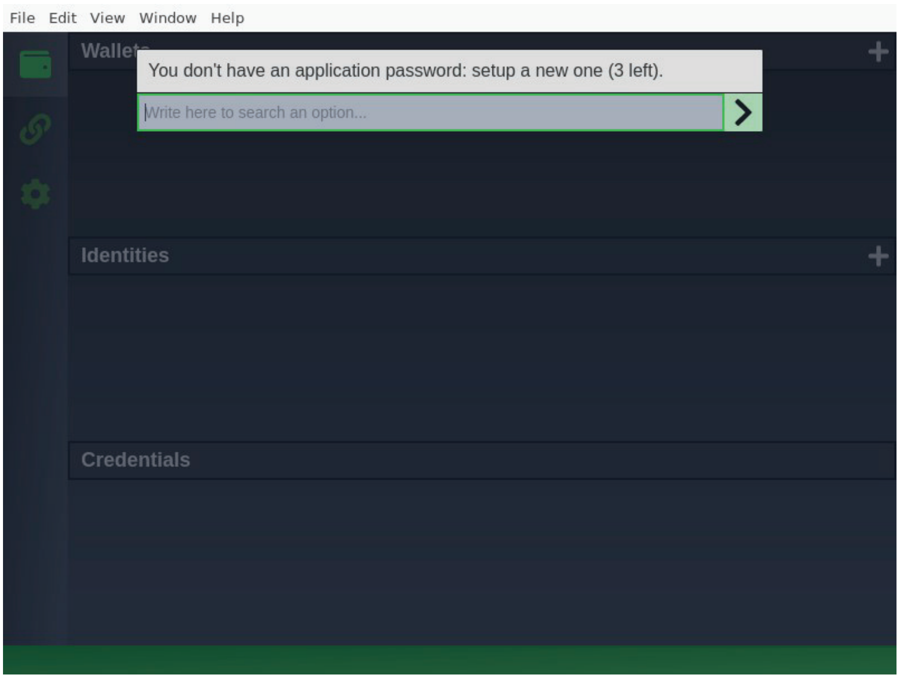
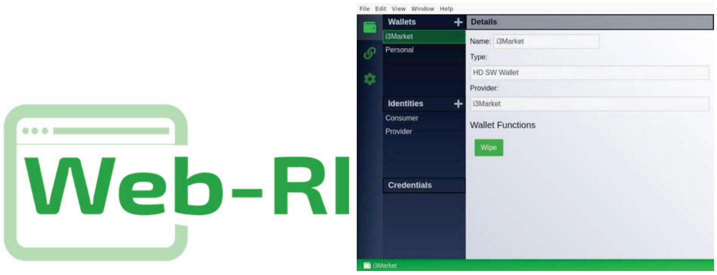

You don't have an application password: setup a new one (3 left).

Figure 5.12 Creating a wallet 1/3.

Create a wallet named i3Market, type HD SW Wallet, and i3Market network – see Figure 5.13.

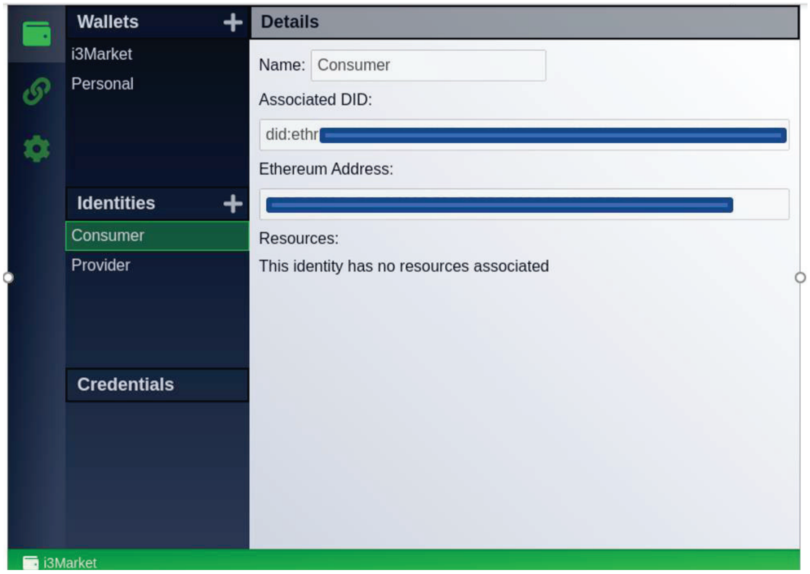


Name: i3Market
Type:
HD SW Wallet
Provider:
i3Market
Wallet Functions

Figure 5.13 WEB-RI interface.

5.7 Creating a Wallet 2/3

Create a consumer and/or provider identity (right-click over the i3Market wallet) – Figure 5.14:

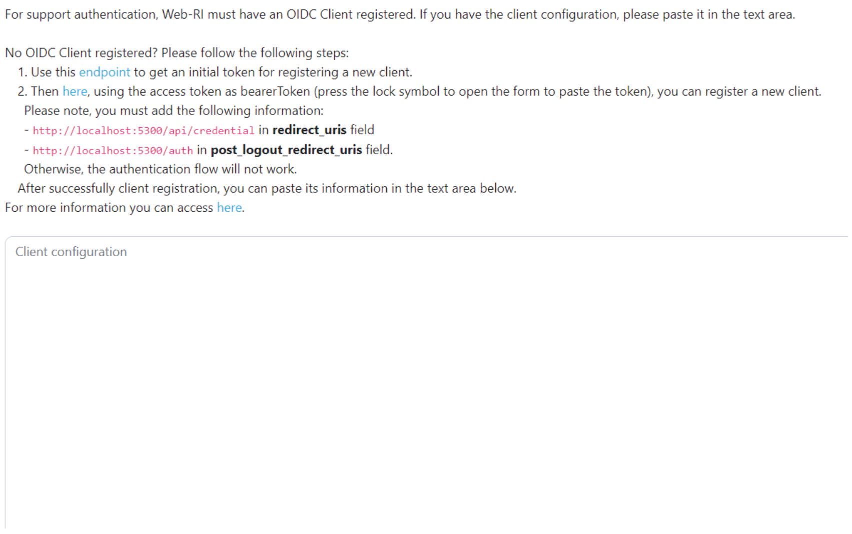


Name: Consumer
Associated DID:
did:ethr
Ethereum Address:
Resources:
This identity has no resources associated

Figure 5.14 Creating a wallet 3/3.

5.8 Register a New OIDC Client

Access to your local instance of WEB-RI (i3-MARKET GUI) available in http://localhost:5300/ and you will be able to see what is shown in Figure 5.15:



For support authentication, Web-RI must have an OIDC Client registered. If you have the client configuration, please paste it in the text area.

No OIDC Client registered? Please follow the following steps:

1. Use this endpoint to get an initial token for registering a new client.
2. Then here, using the access token as bearerToken (press the lock symbol to open the form to paste the token), you can register a new client.
   Please note, you must add the following information:
   - `http://localhost:5300/api/credential` in **redirect_uris** field
   - `http://localhost:5300/auth` in **post_logout_redirect_uris** field.

   Otherwise, the authentication flow will not work.

   After successfully client registration, you can paste its information in the text area below.

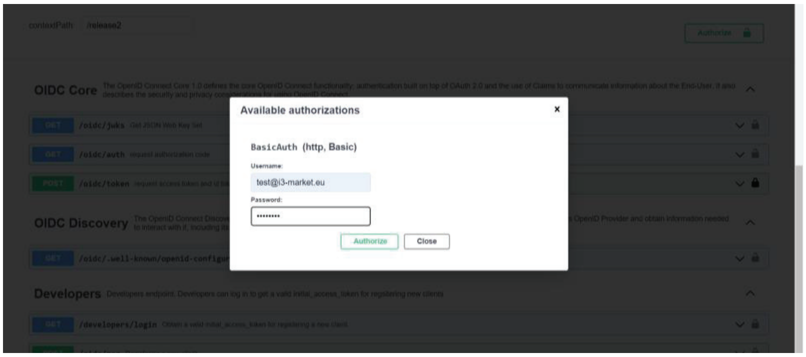
For more information you can access here.

Client configuration

Figure 5.15 OIDC client configuration.

Note: The OIDC client configuration is automatically done from the WEB-RI. Figure 5.16 enables the interaction directly through the SDK-RI or SDK-core must do it by following the next steps.

No OIDC client registered? Please follow the following steps:



Available authorizations

BasicAuth (http, Basic)

Username: test@i3-market.eu

Password: ••••••••

Authorize Close

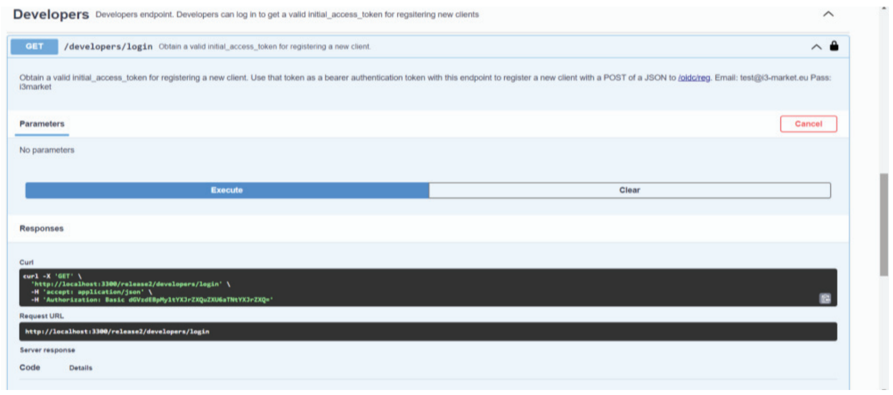
Figure 5.16 Registering an OIDC Client 1/4.

Ask your i3-MARKET admin for your corresponding “i3-MARKET OpenID Connect Provider API”[1] (by default, each instance of i3-MARKET

[1] And endpoint similar to: https://XXXX.i3-market.eu/release2/api-spec/ui/#/Developers/get_release2_developers_login

has its own provider) endpoint to get an initial token for registering a new client (authorize green button).

Try logging in and get *initialAccessToken* as shown in Figure 5.17.



Developers Developers endpoint. Developers can log in to get a valid initial_access_token for regsitering new clients

GET /developers/login Obtain a valid initial_access_token for registering a new client.

Obtain a valid initial_access_token for registering a new client. Use that token as a bearer authentication token with this endpoint to register a new client with a POST of a JSON to /oidc/reg. Email: test@i3-market.eu Pass: i3market

Parameters

No parameters

Responses

Curl

```
curl -X 'GET' \
  'http://localhost:3300/release2/developers/login' \
  -H 'accept: application/json' \
  -H 'Authorization: Basic dGVzdEBpMy1tYXJrZXQuZXU6aTNtYXJrZXQ='
```

Request URL

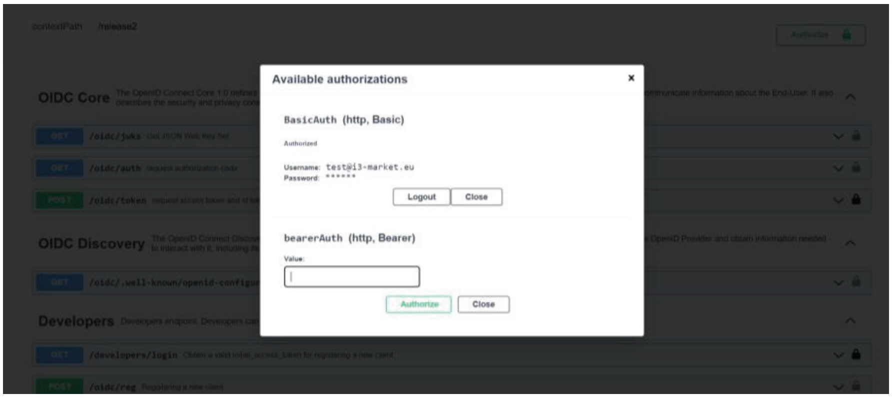
```
http://localhost:3300/release2/developers/login
```

Server response

Code Details

Figure 5.17 Registering an OIDC client 2/4.

Use *initialAccessToken* as *bearerAuth* as shown in Figure 5.18.



Available authorizations

BasicAuth (http, Basic)

Authorized

Username: test@i3-market.eu
Password: ******

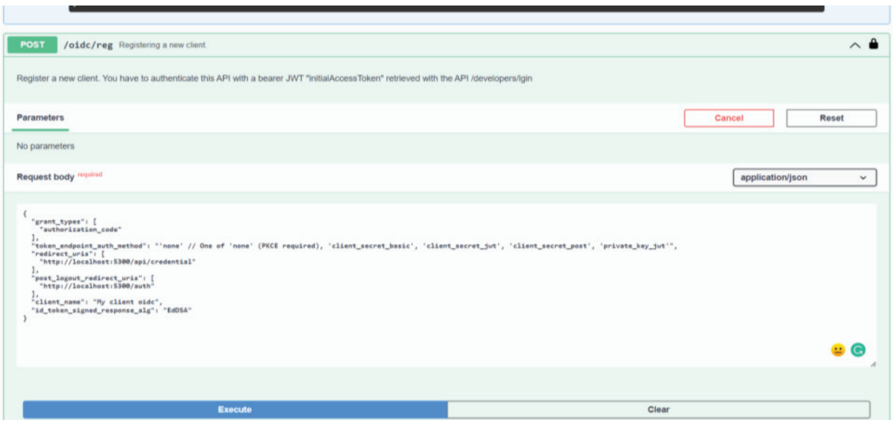
bearerAuth (http, Bearer)

Value:

Figure 5.18 Registering an OIDC client 3/4.

Then here, using the access token as bearerToken (press the lock symbol to open the form to paste the token) – see Figure 5.19 – and you can register a new client. Please note that you must add the following information:

- http://localhost:5300/api/credential in redirect_uris field
- http://localhost:5300/auth in post_logout_redirect_uris field


POST /oidc/reg Registering a new client

Register a new client. You have to authenticate this API with a bearer JWT "initialAccessToken" retrieved with the API /developers/lgin

Parameters

No parameters

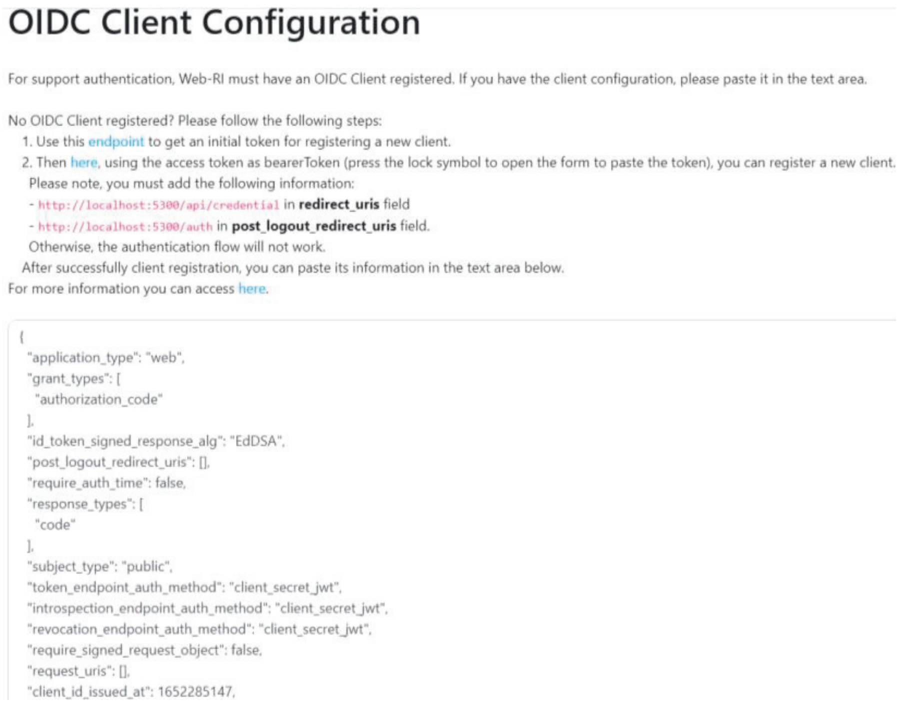
Request body required

application/json

```
{
  "grant_types": [
    "authorization_code"
  ],
  "token_endpoint_auth_method": "'none' // One of 'none' (PKCE required), 'client_secret_basic', 'client_secret_jwt', 'client_secret_post', 'private_key_jwt'",
  "redirect_uris": [
    "http://localhost:5300/api/credential"
  ],
  "post_logout_redirect_uris": [
    "http://localhost:5300/auth"
  ],
  "client_name": "My client oidc",
  "id_token_signed_response_alg": "EdDSA"
}
```

Figure 5.19 Registering an OIDC client 4/4.

After successful client registration, you can paste the returned information in the text area in Figure 5.20.


OIDC Client Configuration

For support authentication, Web-RI must have an OIDC Client registered. If you have the client configuration, please paste it in the text area.

No OIDC Client registered? Please follow the following steps:

1. Use this endpoint to get an initial token for registering a new client.
2. Then here, using the access token as bearerToken (press the lock symbol to open the form to paste the token), you can register a new client.
   Please note, you must add the following information:
   - `http://localhost:5300/api/credential` in **redirect_uris** field
   - `http://localhost:5300/auth` in **post_logout_redirect_uris** field.
   Otherwise, the authentication flow will not work.
   After successfully client registration, you can paste its information in the text area below.

For more information you can access here.

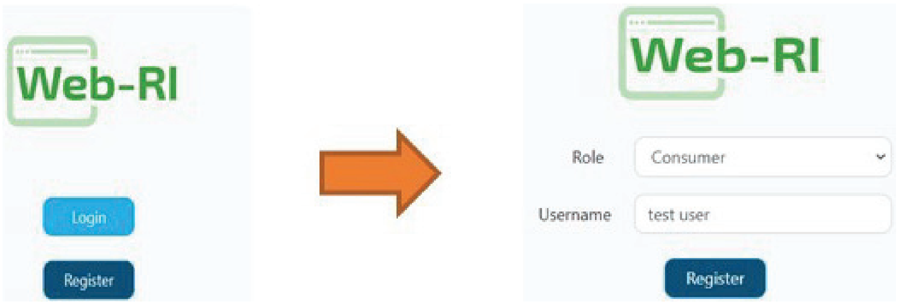
```
{
 "application_type": "web",
 "grant_types": [
  "authorization_code"
 ],
 "id_token_signed_response_alg": "EdDSA",
 "post_logout_redirect_uris": [],
 "require_auth_time": false,
 "response_types": [
  "code"
 ],
 "subject_type": "public",
 "token_endpoint_auth_method": "client_secret_jwt",
 "introspection_endpoint_auth_method": "client_secret_jwt",
 "revocation_endpoint_auth_method": "client_secret_jwt",
 "require_signed_request_object": false,
 "request_uris": [],
 "client_id_issued_at": 1652285147,
```

Figure 5.20 OIDC client registered.

Generate credentials for the consumer/provider identity:

Start the authentication workflow from local WEB-RI instance by following the steps illustrated in Figures 5.21– 5.28

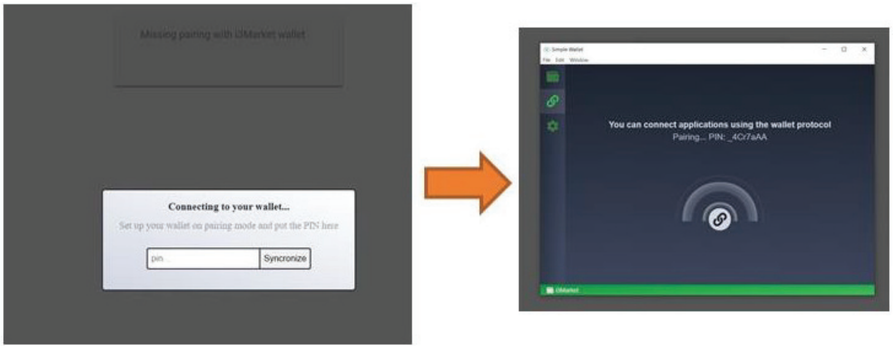
Provide a username for consumer role:



Role: Consumer
Username: test user

Figure 5.21 Username screen.

Wallet pairing:



Missing pairing with i3Market wallet

Connecting to your wallet...

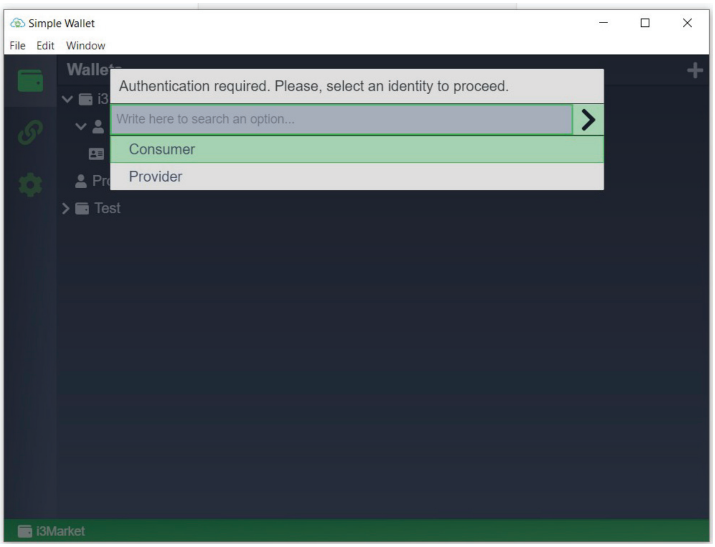
Set up your wallet on pairing mode and put the PIN here

You can connect applications using the wallet protocol

Pairing... PIN: _4Cr7aAA

Figure 5.22 Pairing wallet.

Select wallet identity:



Authentication required. Please, select an identity to proceed.

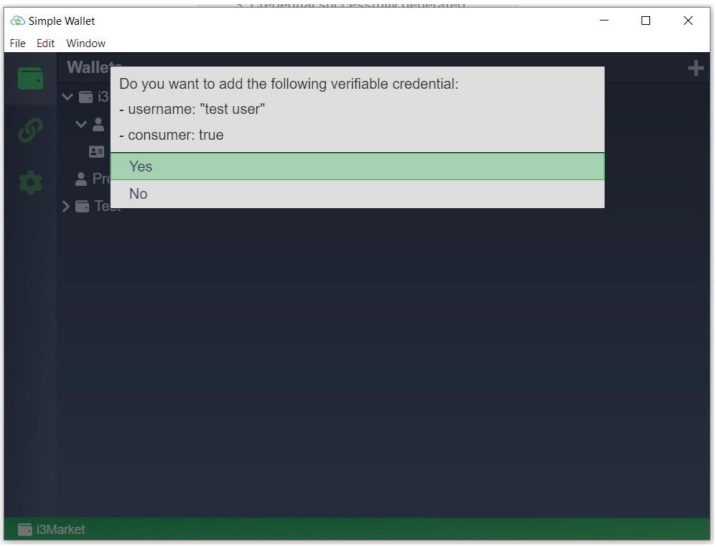
Write here to search an option...

Consumer

Provider

Figure 5.23 Configuring wallet 1/2.

Add Verifiable Credentials to the wallet:



Do you want to add the following verifiable credential:
- username: "test user"
- consumer: true

Yes

No

Figure 5.24 Configuring wallet 2/2.

Login using credentials generated previously:

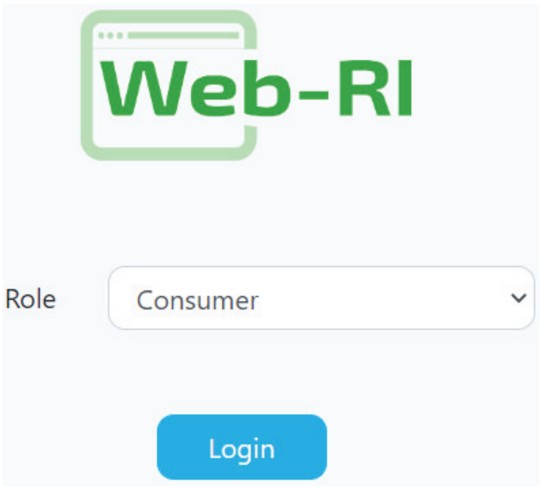
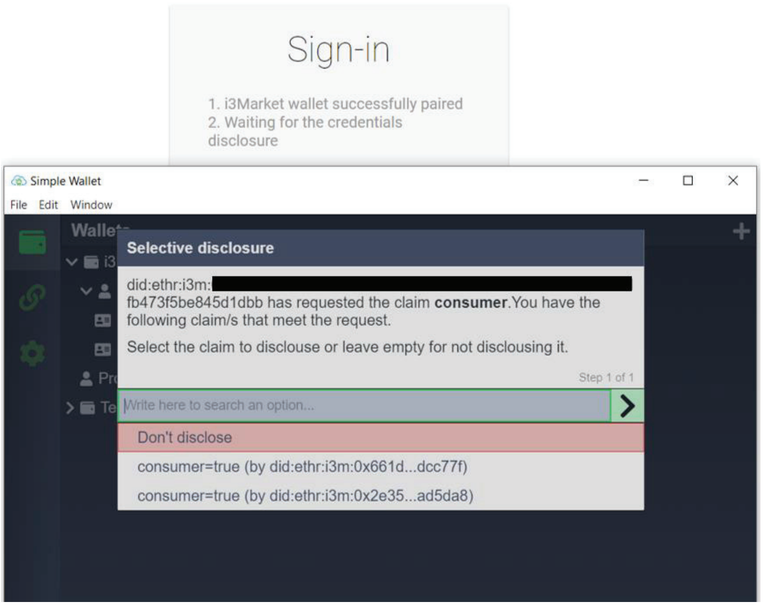


Figure 5.25 Login in WEB-RI.

Selective disclosure:


Sign-in

1. i3Market wallet successfully paired
2. Waiting for the credentials disclosure

Selective disclosure

did:ethr:i3m:[redacted] fb473f5be845d1dbb has requested the claim **consumer**. You have the following claim/s that meet the request.

Select the claim to disclose or leave empty for not disclosing it.

Step 1 of 1

Write here to search an option...

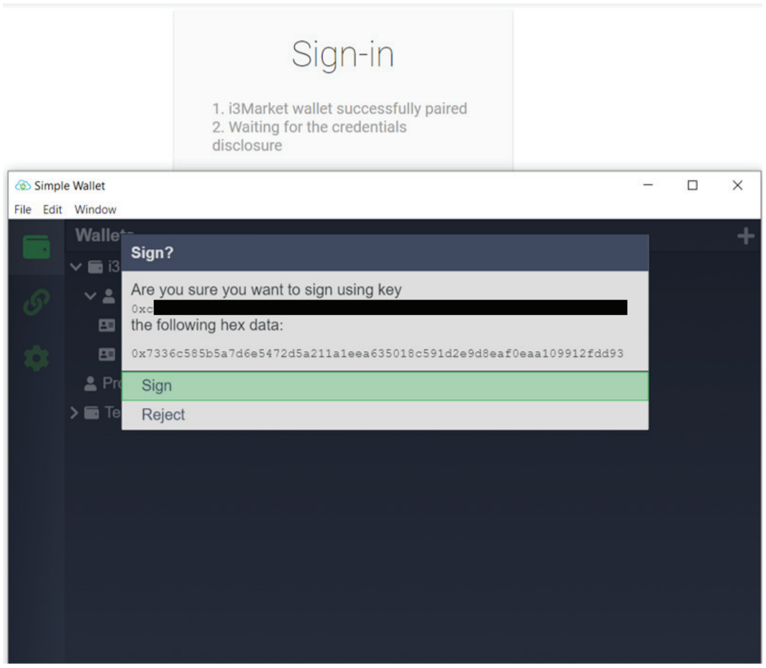
Don't disclose

consumer=true (by did:ethr:i3m:0x661d...dcc77f)

consumer=true (by did:ethr:i3m:0x2e35...ad5da8)

Figure 5.26 Selective disclosure.

Sign:


Sign-in

1. i3Market wallet successfully paired
2. Waiting for the credentials disclosure

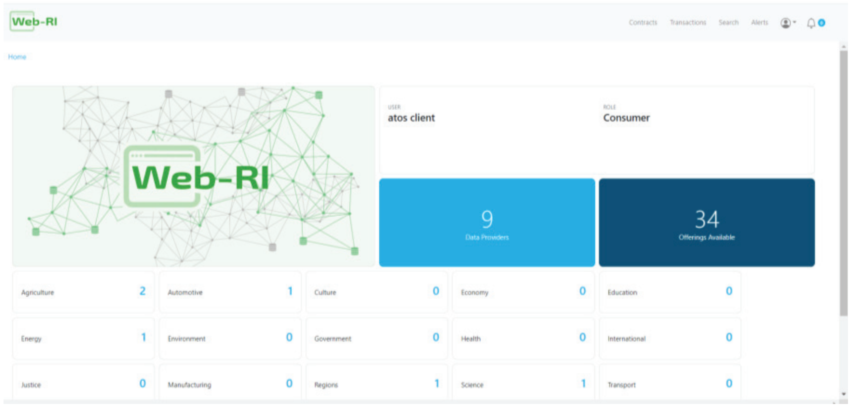
Sign?

Are you sure you want to sign using key
0xc[redacted]
the following hex data:

```
0x7336c585b5a7d6e5472d5a211a1eea635018c591d2e9d8eaf0eaa109912fdd93
```

Figure 5.27 Signing with the wallet.

Access finally to the GUI of Web-RI:



USER atos client

ROLE Consumer

9 Data Providers

34 Offerings Available

Agriculture	2	Automotive	1	Culture	0	Economy	0	Education	0
Energy	1	Environment	0	Government	0	Health	0	International	0
Justice	0	Manufacturing	0	Regions	1	Science	1	Transport	0

Figure 5.28 Accessing WEB-RI.

5.9 SDKs

Technical requirements:

The current subsection contains a set of SDK requirements that have been collected for releases 2 and 3. Most of them have been extracted from D2.5 [3]; meanwhile, the other ones are the result of deepening in the last iterations of SDK elicitation process.

SDK-core:

The SDK-core is built using SDK-generator REST API and an Ansible playbook in charge of generating all the client stubs for Backplane API (semantic engine, notification manager, and smart contract manager), OIDC, VC, and Data Access API encapsulated into the SDK-core Java/JavaScript library.

SDK-core specification:

Backplane API SDK: The main goal of the SDK is boasting the Backplane API to create applications for the i3-MARKET platform. It will assist the data marketplaces and stakeholder developers with a set of tools, examples, and documentation, which will reduce the developing effort to be part of the i3-MARKET ecosystem. The Backplane API SDK content is divided into different logical modules, which correspond to each of the i3-MARKET modules integrated in the Backplane API. In the following, the different modules identified for the first version of the requirement specification can be seen:

- User-centric authentication SDK
- Cloud Wallet SDK module
- Data access SDK module
- Standard payments SDK module
- Tokenization SDK module

Enhanced Backplane API SDK: For some cases, the SDK will complete the Backplane API services with its own logic to support the developers in the use of the i3-MARKET capabilities. These will be done through a set of workflows.

Automatically build Backplane API SDK: In addition to the inner SDK functionality, i3-MARKET will provide mechanisms to automatically build the SDK component and it will be offered in different programming languages.

SDK-core implementations:

The SDK-core implementation is based on the usage of SDK-generator, and it is described in detail in the following subsections.

Core technology:

The SDK-core is supported by means of (a) the SDK-generator REST API and (b) an Ansible playbook in charge of generating:

- An SDK-core Java artifact that contains client stub for Backplane API (semantic engine, notification manager, and smart contract manager), OIDC (OpenID Connect), VC (Verifiable Credentials), and data access API.
- An SDK-core JavaScript artifact contains client stub for Backplane API (semantic engine, notification manager, and smart contract manager), OIDC, VC, and data access API.

SDK-generator:

The SDK-generator is the main pillar of the SDK-core. The SDK-generator is based on SDK as a service approach. SDK-generator aims to automatically generate the client stubs needed to interact and consume all the functionalities exposed in a REST API. The SDK as a service approach is shown in Figure 5.29.

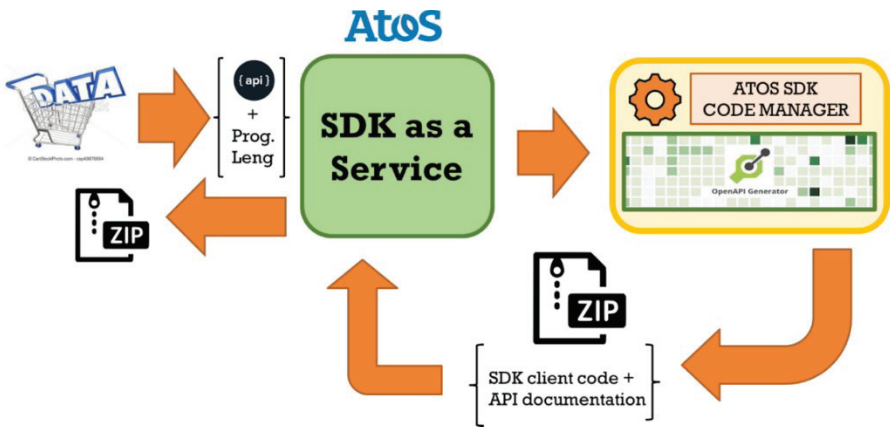


Figure 5.29 SDK-generator approach.

The workflow behind SDK-generator is based on the provision of a programming language specification next to an OAS file and making use of the OpenAPI generator server, which is able to produce as output SDK client stubs next to associated documentation about how to use it.

The languages supported by the SDK-generator are shown in Figure 5.30 as part of the SDK as a service configuration.

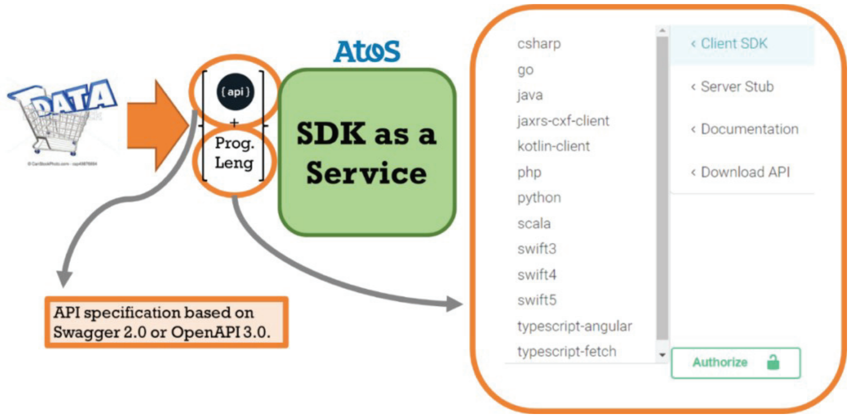


Figure 5.30 SDK generator supported programming languages.

Continuous integration and delivery:

The SDK-core artifact is automatically provided by means of a CI/CD pipeline based on Ansible AWX. A conceptual view of SDK-core pipeline is shown in Figure 5.31.

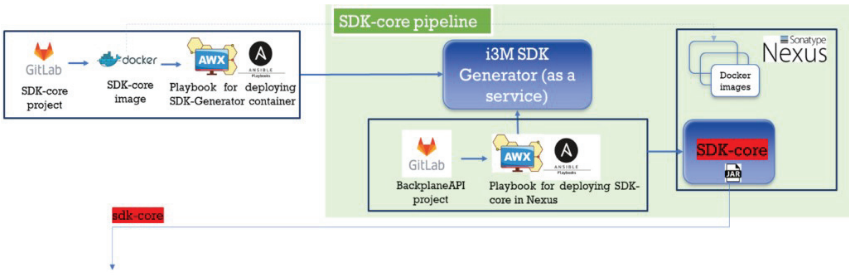


Figure 5.31 SDK-core CI/CD pipeline.

As initial step in the pipeline, the SDK-core artifact is triggering the compilation and deployment of a new version of the SDK-generator once a commitment into master branch of SDK-generator project happens. As a second step (represented as green area in Figure 5.31 - SDK-core CI/CD pipeline), the generation and publishing of a new version of the SDK-core artifact is triggered by using a new version of backplane API which is deployed each time the SDK-core artifact is triggered. The CI/CD behind backplane API includes a triggering to the SDK-core pipeline. In this way, SDK-core covers a set of tasks mainly in charge of generating SDK-core artifacts for Java and JavaScript versions taking a set of relevant OAS files associated with the following artifacts:

- Backplane API (including semantic engine, notification manager, and smart contract manager)
- OIDC API
- Verifiable Credentials API
- Data access API

Finally, the pipeline includes a couple of tasks in charge of publishing the generated Java and JavaScript versions of SDK-core into i3-MARKET Nexus repository.

SDK-core installation:

SDK-core is a Java/JavaScript library that is installed by simply importing from i3-MARKET official Nexus repository.

SDK reference implementation (SDK-RI):

The current section reports on SDK-reference implementation specification, its implementation, and, finally, its deployment and installation.