Build documentation

Requirements

To run the connectors on your own machine, installing Docker is enough. If you don't want to use Docker, make sure that you have the following packages installed:

Package	Version
JDK	17
Gradle	8.7
jq	1.7.1
npm	10.7
node	22.2

Build process

Docker usage

To run the code using docker, use the following command in the `src` folder:

```
sudo docker compose up
```

Note: If you are using macOS, you might have to modify the `config.json` file:

1. Go to `~/.docker/config.json`.
2. Change the `credsStore` value from `desktop` to `osxkeychain`.

Alternatively you may:

1. Go to `sudo vi ~/.docker/config.json`.
2. Change `credsStore` to `credStore`.

Running the connectors locally

To run the connectors without using Docker, you have to use four different terminals. Use the following commands in separate terminals:

Company connector

In the first terminal, use the following command to build Gradle project and run the company connector:

```
./gradlew connector:build

java -Dedc.keystore=resources/certs/cert.pfx \
-Dedc.keystore.password=123456 \
-Dedc.vault=resources/configuration/company-vault.properties \
-Dedc.fs.config=resources/configuration/company-configuration.properties \
-jar connector/build/libs/connector.jar
```

Tax advisor connector

In the second terminal, use the following command to run the tax advisor connector:

```
java -Dedc.keystore=resources/certs/cert.pfx \
-Dedc.keystore.password=123456 \
-Dedc.vault=resources/configuration/tax_advisor-vault.properties \
-Dedc.fs.config=resources/configuration/tax_advisor-configuration.properties \
-jar connector/build/libs/connector.jar
```

Bank connector

In the third terminal, use the following command to run the bank connector:

```
java -Dedc.keystore=resources/certs/cert.pfx \
-Dedc.keystore.password=123456 \
-Dedc.vault=resources/configuration/bank-vault.properties \
-Dedc.fs.config=resources/configuration/bank-configuration.properties \
-jar connector/build/libs/connector.jar
```

Running the web app

Run the app in the forth (main) terminal:

```
cd frontend_socket/international-dataspace-station
npm run dev
```

After this, you can access the app at `https://localhost:3000`.

Establishing connection for data exchange

Send the following HTTP requests to establish a connection between different connectors to be able to exchange data (replace `{{provider port}}`/`{{consumer port}}` with the corresponding ports on which the connector that provides/consumes data is running):

1. Register data plane

```
curl -H 'Content-Type: application/json' \
-d @resources/dataplane/register-data-plane-provider.json \
-X POST "http://localhost:{{provider port}}/management/v2/dataplanes" -s | jq
```

2. Create an asset

```
curl -d @resources/create-asset.json \
-H 'content-type: application/json' \
http://localhost:{{provider port}}/management/v3/assets \
-s | jq
```

3. Create a policy

```
curl -d @resources/create-policy.json \
-H 'content-type: application/json' \
http://localhost:{{provider port}}/management/v2/policydefinitions \
-s | jq
```

4. Create a contract definition

```
curl -d @resources/create-contract-definition.json \
-H 'content-type: application/json' \
http://localhost:{{provider port}}/management/v2/contractdefinitions \
-s | jq
```

5. Fetch catalog

```
curl -X POST "http://localhost:{{consumer port}}/management/v2/catalog/request" \
-H 'Content-Type: application/json' \
-d @resources/fetch-catalog.json -s | jq
```

6. Negotiate contract

Replace the `{{contract-offer-id}}` placeholder in `negotiate-contract.json` with the contract offer id you found in the catalog at the path `dcat:dataset.odrl:hasPolicy.@id`:

```
curl -d @resources/negotiate-contract.json \
-X POST -H 'content-type: application/json' \
http://localhost:{{consumer port}}/management/v2/contractnegotiations \
-s | jq
```

7. Get contract agreement id

Replace `{{id}}` with the contract negotiation id from the consumer terminal:

```
curl -X GET \
"http://localhost:{{consumer port}}/management/v2/contractnegotiations/{{id}}" \
--header 'Content-Type: application/json' \
-s | jq
```

The connectors have now been configured successfully and are ready to be used.