8

SDK-RI Specification

8.1 Objectives

The SDK reference implementation, or SDK-RI, has these specific objectives:

- Provide the mechanisms in terms of SW pieces for testing the i3-MARKET Backplane services/artifacts.
- Follow the approach SDK-RI as a service: SDK-RI will be a set of services needed for simulating an i3-MARKET-ized data marketplace behaviour.
- SDK-RI will let the pilots check this reference implementation as a guide/example for developing their own integration with i3-MARKET.
- Context: SDK-RI contextualization was already introduced in section 6.2 as part of the SDK-core.

8.2 Technical Requirements

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

8.3 SDK Reference Implementation

The SDK-RI implementation is based on Java and Swagger framework, and the next subsections are focusing on the update provided during R2 and R3 developments. The SDK-RI was first released as a web app deployed within Jetty and encapsulated in a Docker container then later in R2 and R3 updated with Java and Swagger.

8.4 Core Technology

In an initial stage of SDK-RI implementation, the technology options presented in Figure 8.1 – Implementation technologies for SDK-RI – were considered:

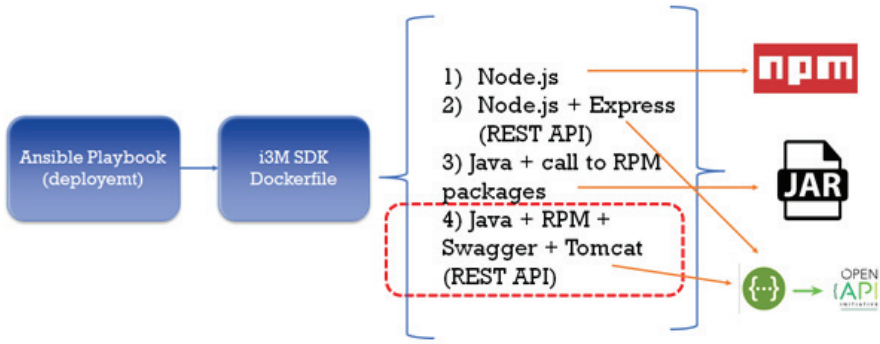

Figure 8.1 SDK-RI Implementation Technologies Used.

To sum up, the candidate technologies to support the implementation of SDK-RI were the following:

- Node.js
- Node.js + Express
- Java + RPM
- Java + Swagger + Tomcat

Finally, option 4 was selected but substituting Jetty for Tomcat as web application server. Therefore, we can conclude by saying that SDK-RI is a web app deployed within Jetty and encapsulated in a Docker container.

8.5 Continuous Integration and Deployment

The SDK-RI 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 8.2 – SDK-RI pipeline.

As initial stage, the SDK-RI is imported as a library in the last version of the SDK-core published in i3-MARKET Nexus maven repository. As a second stage, once a commit is done into the master branch of SDK-RI GitLab project, a compilation and deployment of a new version of SDK-RI is carried out.

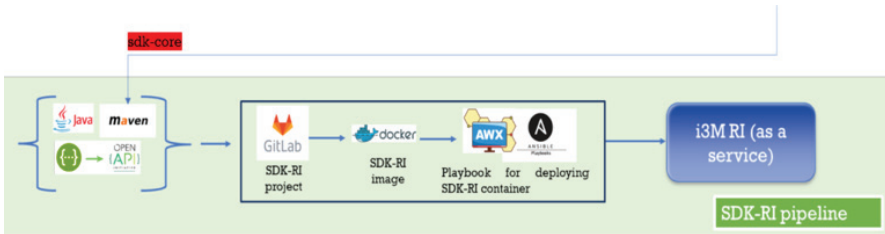

Figure 8.2 SDK-RI pipeline based on Ansible AWX.

SDK-RI installation:
The setup instructions and Docker-based deployment of SDK-RI is covered in detail in the following subsections.

Setup:
Clone the repository and download the dependencies:
git@gitlab.com:i3-market/code/sdk/i3m-sdk-reference-implementation.git

Running the SDK-RI with Docker:
Use Docker to run the SDK-RI. To do so, follow the same setup instructions as above.

Then, just build your SDK-RI project nd run it using the jetty images as follow:

SDK-RI container is built over a Jetty image and the SdkRefIMpl war file is deployed into Jetty.

Finally, just go to http:/$deploy_host/SdkRefImpl for accessing SDK-RI REST API.

Configuring and using SDK-RI

To configure SDK-RI instance, the following steps should be covered:

- The marketplace will have all the common services exposed in an SDK-RI/endpoint.

Each marketplace end-user, which pursues making use of the SDK-RI, should configure the SDK-RI by means of:

- pointing to the Backplane endpoint(s) hosted in a concrete i3-MARKET node (i.e., Backplane API node1, OpenID Connect Provider API node1, Verifying and Credential service API node1);
- pointing to the wallet endpoint hosted locally.

This configuration should be defined in the SDK-RI properties file placed at ‘‘src/resources/sdk_ri_config.properties’’.

The internal workflow covered by the SDK-core/RI playbook is shown in Figure 8.3.

Annex B (SDK-core/RI playbook) contains the last version of Ansible playbook that supports the generation of the SDK-core/RI for the *final* release (or R3).

Figure 8.3 SDK-core/RI playbook internal workflow.