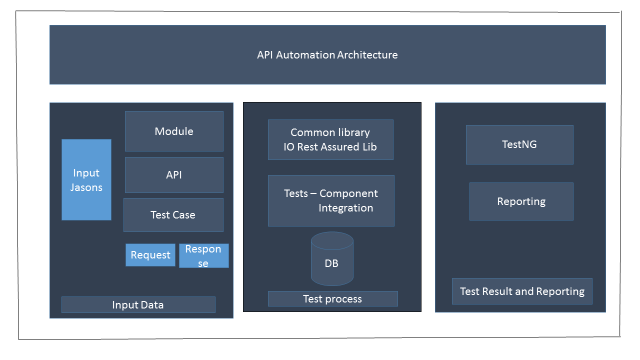
**1. Overview**

API testing will be carried out in 2 stages, both via Test Automation.

1. Soap UI automation - This approach is mainly to catch up with the backlog and disclose bugs sooner, however this approach has disadvantages when the APIs undergo changes
2. Restassured API test framework development and test automation - This will be a more generic framework, that is both modular and comes with less cost of maintenance

# 2. Architecture view

Rest Assured with io DSL follows the below architecture view.



**Tools and Technologies**

IO Rest Assured DSL

TestNG

Java/J2EE

Maven

Eclipse Editor

**The framework consists majorly 3 Elements/parts as below:**

Test Data

Test Process/Execution

Results/Report

**3.Directory Structure**

Io.mosip.modulename.tests - Contains test scripts

Io.mosip.service – Library functions

Io.mosip.util – Utility functions for data/report handling

Io.db.dbaccess – DB Access

Src/test/resources – Test inputs and configuration files

# 4. Building Tests

## 4.1 Test Data

Test Data is maintained under the path src/test/resources. Every module's data is saved separately from other module with folder name as module name (say Kernel). Every api under specific module is saved as folder name. All test cases belonging to an api are saved with separate folders; the name of folder being the name of the test case. Each test case folder has 2 jsons; one for request.json and another referring expected response.json. It looks as below: src/test/resources/moduleName/apiName/testCaseName/Request.json src/test/resources/moduleName/apiName/testCaseName/Response.json

Example: src/test/resources/Kernel/FetchDevice/invalid\_deviceType\_blank/Request.json src/test/resources/Kernel/FetchDevice/invalid\_deviceType\_blank/Response.json

## 4.2 Test Scenario

Test Cases are formed around api's behavior and its attributes. Every api will have valid and invalid scenarios with varying combination of attributes of the api. First success scenario test cases are formed by keeping all attributes as valid. Then error scenarios are formed by updating a field as invalid one at at time. Each folder name under specific api represents an individual test case.

## 4.3 Test Execution and assert

IO Rest Assured methods (POST, GET, PUT, and DELETE) used to run the requests. These methods saved under Common Library so that same methods are re-used. Tests are executed using io rest assured DSL libraries. All methods are implemented under CommonLibrary.java and ApplicationLibrary.java file contains generic methods to accept input body and resource uri. These files can be located under io.mosip.util package.

All tests are maintained module wise; each class under tests represent an api specific to the module. The name of the java class is same as api name. The same java class is having test methods to run multiple test data combination for an api. The directory structure is: io.mosip.kernel.tests io.mosip.preregistration.tests io.mosip.authenticatiion.tests io.mosip.registrationprocessor.tests

Example test class file: io.mosip.kernel.test.AuditLog.java

After getting actual response from the service, the actual response body is compared with expected response body by using hashcode and then json to json comparison by removing dynamic elements from both response json. Response files are converted to Json Object using Json Mappers and then Object to Object is compared. The code for assert is present under: io.mosip.service.AssertResponse java file.

## 4.4 Test Results/Reportig

Based on the assert function output a test is decided as PASS or FAIL and then written to TestNG default report emailable.html report. This report is found under test-output/emailable-report.html. The same report will be circulated to other audience after pipeline QA build.

## 4.5 Creating Component Tests

Component level tests are built around api. Input test data is fetched from test/resources/modulename/apiname/testcase level. The request is sent to server using IO Rest assured DSL (Domain Specific Libraries). Then Actual Response from the IO Rest assured and Expected Response from the resource folder is compared to check against the validity. If any dynamic elements are there in the response then they are deleted before comparison. Based on the assertion result, specific test is decided PASS or FAIL. The result is taken further to TestNG report

## 4.6 Creating Integration Tests

Individual methods are written for all the apis under a module. The inputs for these methods are taken from the already existing resources. Integration scenario is construced by calling individual methods as below.

Scenario: Delete Document for an application

Methods called are:

CreateApplication()

DocumentUpload()

FetchRegistrationCentre()

BookingAppointment()

DeletePreRegistrationByUser()

# 5. Running the Tests

To run the tests we need to provide the environment as the input to say on which environment the tests need to be run.

Following are the ways to run.

1. As an individual suite from testing.xml file
2. As a whole module from pom.xml file

Choose the respective module testing.xml (Ex: testngPreRegistration.xml)

RighClick>select runas>RunConfiguration>under Arguments pass one of these based on the environment

-Denv.user=qa

-Denv.user=integration