TestCaseCollection: A group of test cases, created by the user in the xlsx input file is called a TestCase Collection. TestCaseCollection.jar is used to create TestCaseCollections. This jar has 2 functionalities: 1) create 2) fetch, and 2 input files: 1) Property file 2) Excel file.

create: This will create TestCaseCollection. (Note: The created TestCaseCollection will be stored in QAMill DB) which has to be fetched later by fetch operation. Name of the TestCaseCollection should be provided in property file and name of the testcases should be provided in the excel file. Create operation will read the following information from the property file MosiptestFrameworkConf.property provided in the environment variable with the name MosipTestFrameworkRT:

* Operation: Should be “create” (without quotes).
* TestCaseCollectionsName: Name of the TestCaseCollection.
* TestCaseExcelFilePath: Path of the excel file having hints to create TestCaseCollection.

Note: Test cases should be unique. They however are not tied tightly to the Collection name. Therefore same test case “name” cannot be provided twice for TestCase. Create operation fails in such case. Also TestCaseCollection name is unique as well.

Fetch Operation: This will fetch the specified TestCaseCollection (along with corresponding TestCases) to the specified path. Following keys are considered

1. operation: Should be “fetch” (without quotes).
2. TestCaseCollectionsPath: Path where all TestCaseCollections are stored. If this path does not exist, then it will be created.
3. TestCaseCollectionsName: Name of the TestCaseCollection folder. This folder will be created in TestCaseCollectionsPath. If folder already exist, its contents will be overwritten.

The fetched test cases will be stored in the TestCaseCollections folder in the path specified in the TestCaseCollectionsPath key in property file. A folder will be created for each testcase with the name being the name of the test case. TestCase folder will contain 3 files; request.json, response.json and stitching.json. request.json and response.json files will have requestbody and responsebody, whereas stitching.json file will have informations like URL, RequestMethodtype, Params, Headers, and stitching information for request & expectedresponse body. TestCaseCollections folder will have one file, called TestCaseExecutionSequence which will contain the names of the test cases in the execution sequence. (Note: Operation “execute” in TestCaseExecutionframework which is explained later will use this file and executes the testcases accordingly)

Providing hints in the Excel file for “create” operation:

**#**: Serial Number

**Type**: Always enter the string “ENDPOINTWITHHINTS-TYPE.TEST”.

**API Execution Order:** Start from 1. Numbering should be in serial order.

**API Name:** Name of the API provided in the ServiceRequestMethod. URL of the API, as well as requestMethodType, Request JSON & Response JSON will be fetched (from QAMill DB) based on API name.

**TestCase Name:** Name of the testcase prepared from that row. This should be unique across all testcasecollections.

**TestCase Type**: Test Strategy / Plan specific

**TestCase Significance**: Test Strategy / Plan specific

**Expected Response**: Expected Response JSON for the testcase. If this is empty, default responsebody of the api will be picked otherwise JSON from ErrorResponse store whose name matches with the one provided in this cell will be picked.

**Hints**: Changing values in request or expected response json or params or headers.

**Key:**

**Request JSON field:**

**INTERPRETATION**: INTERPRETATION.SUBSTITUTE

**KEYMAJORTYPE**: KEYMAJORTYPE.REQUESTBODY

**KEYMINORTYPE**: KEYMINORTYPE.FULLYQUALIFIEDNAME

**Request Parameter:**

**INTERPRETATION**: INTERPRETATION.SUBSTITUTE

**KEYMAJORTYPE**: KEYMAJORTYPE.REQUESTPARAM

**KEYMINORTYPE**: KEYMINORTYPE.VARIABLE

**Request Headers:**

**INTERPRETATION**: INTERPRETATION.SUBSTITUTE

**KEYMAJORTYPE**: KEYMAJORTYPE.REQUESTHEADER

**KEYMINORTYPE**: KEYMINORTYPE.VARIABLE

**Expected Response JSON field:**

**INTERPRETATION**: INTERPRETATION.SUBSTITUTERESPONSE

**KEYMAJORTYPE**: KEYMAJORTYPE.EXPECTEDRESPONSEBODY

**KEYMINORTYPE**: KEYMINORTYPE.FULLYQUALIFIEDNAME

**Value:**

**Value**: Provide value in the cells. (Note: For numbers, format the cell to Text.)

**ValueDataType**: This can be String, Integer, JSON or JSONArray.

Input can be provided in multiple sheets. Execution order will be based on the “API Execution Order” column. Stitching of APIs is also possible. In this case, instead of providing value directly in the cell, we need to prepend the keyword “DynEPRef---TEST”. Ex: DynEPRef---TEST---1---RESPONSEBODY---id means to fetch id from ResponseBody from the api having execution order 1.

TestCaseExecution: Test Cases in a TestCaseCollection can be executed. The result will be stored in TestCaseExecution folder.

TestCaseExecutionFramework application is to be used to execute test cases. TestCaseCollectionsPath and TestCaseCollectionsName should be provided in the property file so that the corresponding test cases are executed, as per the execution sequence provided in the ExecutionSequence file. Each of these test cases are traversed and executed. RequestBody will be obtained from request.json file and informations like URL, params & headers are obtained from stitching.json of the corresponding test case folder. The actual response and expected response are placed in the corresponding executed testcase folder inside the ExecutedTestCaseCollections folder in the TestCaseExecutionsPath provided in the property file.

Following information need to be provided in the property file:

Operation: “execute” to execute testcases.

TestCaseCollectionsPath: Path from where to read the testcases of the testcasecollection.

TestCaseCollectionsName: Name of the folder, in the above path, containing all test cases.

TestCaseExecutionsPath: Path to where the expected and actual response of the executed testcases of the testcasecollection is to be copied.

TestCaseExecutionsName: Name of the folder containing all executed test cases.

MosipBaseURL: Base URL to execute the testcases.