Solution Framework

Low Code Test Automation Generation

|  |  |  |  |
| --- | --- | --- | --- |
| FrameWork # | Author | Date | Content |
| 0.8 | Danny Saro | 2019-01-30 | Initial framework description |
|  |  |  |  |

Contents

[Objective 2](#_Toc536690716)

[Features 2](#_Toc536690717)

[High-Level Architecture 2](#_Toc536690718)

[Installation & Configuration 2](#_Toc536690719)

[Download 2](#_Toc536690720)

[Configuration 3](#_Toc536690721)

[User Guide 4](#_Toc536690722)

[The Test Specification (Source File) Repository 4](#_Toc536690723)

[Structure of the parent Repository directory 4](#_Toc536690724)

[Structure of a Test Specification directory 5](#_Toc536690725)

[A Test Specification 6](#_Toc536690726)

[The Request file 6](#_Toc536690727)

[The Validation file 6](#_Toc536690728)

[Currently implemented validation types 6](#_Toc536690729)

[Currently implemented filter types 7](#_Toc536690730)

[The Test Suite Specification 7](#_Toc536690731)

[A Test Generation Run 7](#_Toc536690732)

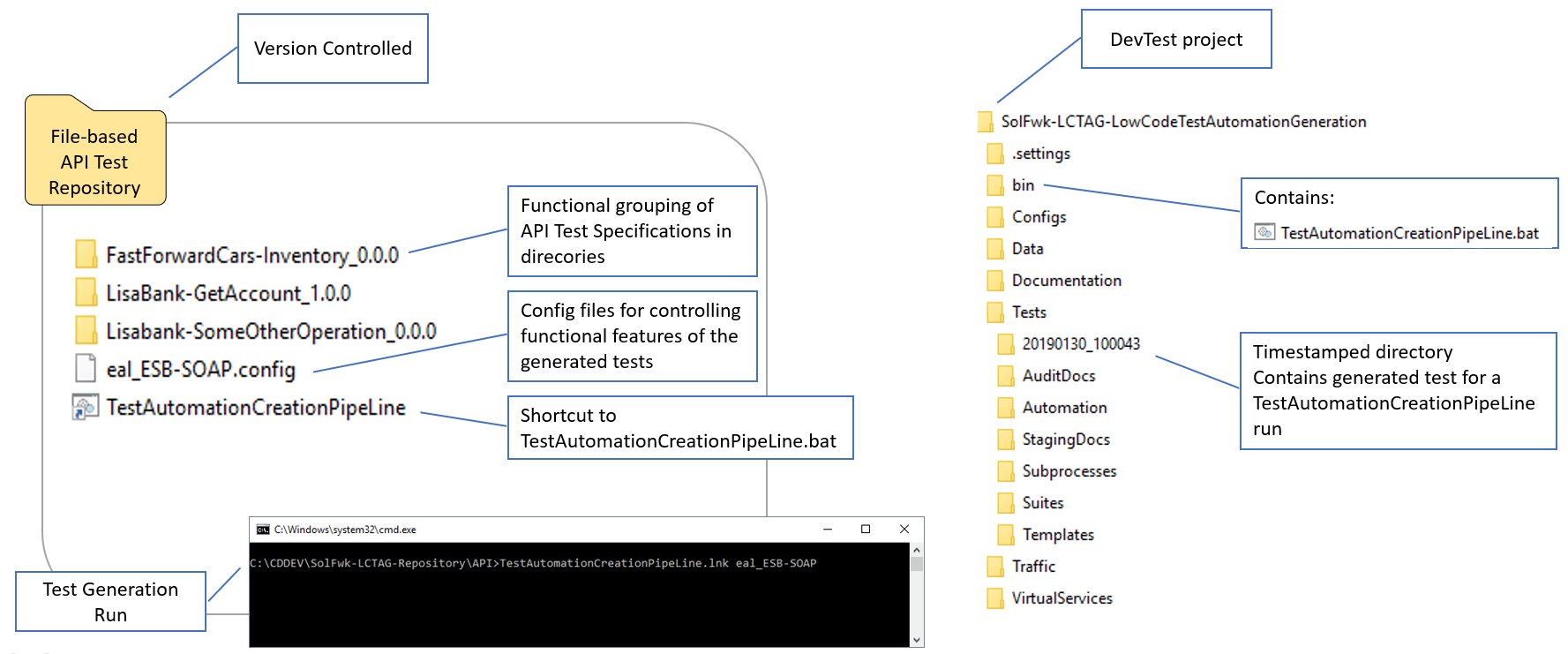
# Objective

* Generate suites of executable tests only from behavior specification
* Test complexity can range from low complexity API tests to medium complexity Business Functional tests

# Features

* Test behavior specification can be maintained in a version controlled test source file repository (recommended)

# High-Level Architecture



# Installation & Configuration

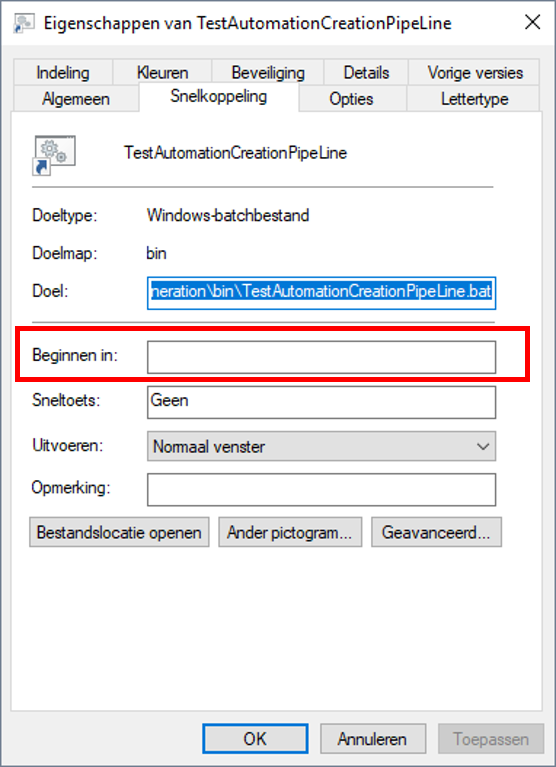
## Download

* Download following 2 directories:
  + <https://github.com/DannySaro/CA-DevTest-Solution-Patterns/tree/master/SolFwk-LCTAG-LowCodeTestAutomationGeneration>
  + <https://github.com/DannySaro/CA-DevTest-Solution-Patterns/tree/master/SolFwk-LCTAG-Repository>
* Each directory can be located independently from the other
* Each directory can be renamed as wanted
* The SolFwk-LCTAG-Repository directory actually contains 2 repositories (currently subdirectories of this parent directory)
  + “API” repository and “VirtualServiceValidation” repository
  + Each of these directories can be located independently from the other (they do not have to reside in this parent directory)
  + Each of these directories can be renamed as wanted

## Configuration

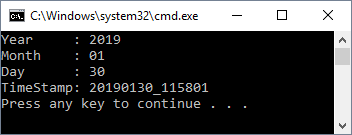
**Create a correct shortcut to TestAutomationCreationPipeLine.bat**

* As mentioned, “API” and “VirtualServiceValidation” are the actual repositories so each contains the shortcut
* For the properties of the shortcut, blank out “Start In” property such that the working directory during a run is the repository directory (and not the batch file’s target directory)



**Validate the timestamp creation commands (and adapt if needed)**

* During a TestAutomationCreationPipeLine.bat run a timestamp is created to allow for storing the generated assets in a unique location. Timestamp creation is done using the “date” command and output of this DOS command is locale specific. Hence depending on your locale it might be necessary to adapt TestAutomationCreationPipeLine.bat
* Directory SolFwk-LCTAG-LowCodeTestAutomationGeneration\bin contains a TimeStamp.bat file that can be used to assess and/or correct the timestamp generation.
  + Run the command
  + Check the output, the timestamp should be correct, there can be NO SPACES in the timestamp



* + If the timestamp is not correct then you can use the TimeStamp.bat file to find the correct statements. You can edit this file at will it is not used otherwise.
* If the timestamp commands needed editing then these adapted commands have to be inserted into the “TestAutomationCreationPipeLine.bat” inside “SolFwk-LCTAG-LowCodeTestAutomationGeneration\bin”

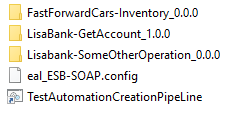
**Adapt “TestAutomationCreationPipeLine.bat”**

* Correct the SET LISA\_HOME=... statement to point to your CA DevTest installation directory
* Correct TimeStamp setting if needed (see above)
* Correct TestRunner command authentication, i.e. the -u and -p parameters, the user must have a minimum of "Test Runner" authorization

# User Guide

## The Test Specification (Source File) Repository

### Structure of the parent Repository directory



**Test Specification Directories**

* Tests (test behavior specifications) that functionally belong together are stored together in a directory, e.g. LisaBank-GetAccount\_1.0.0. Tests in here logically belong to the same Test Suite. Groupings can be as granular or as broad as needed, e.g. all API operations in one directory, or, each API operation in its own directory
* There can be a hierarchy of directories for easier management, e.g. a directory for the API with subdirectories for the Operations
* A generation run will only build tests for the specifications inside one directory, not for the content of its subdirectories

**Config Files**

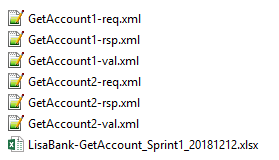
* A config file will be used as a parameter to the TestAutomationCreationPipeLine batch file.
* The needed properties inside this config file will be dictated by the test template that is used during test case generation
* This DevTest .config file must reside as well in the current working directory as in the config directory of the LowCode TestAutomation Generation DevTest project.

**TestAutomationCreationPipeLine shortcut**

* The TestAutomationCreationPipeLine batch file will eventually run an "automation" testcase inside a LowCode TestAutomation Generation DevTest project. The main input for running the "automation" testcase is an excel file that identifies the testcases to build. This is not to be confused with the input to the batch file itself which is a config file. For the excel file see below section on “The Test suite Specification”.
* Parameter to this batch file is a DevTest .config file which must reside ss well in the current working directory as in the config directory of the LowCode TestAutomation Generation DevTest project. The parameter must contain only the config file name, not the .config suffix
* This batchfile will open a Select File Dialog Box (to select the excel identifying the testcases) only showing the filesystem below the working directory where this batch file is started at runtime.

If running this by executing a shortcut located in the testcase repository master document then the "Start in" property of the link has to be blanked out (so the batch file starts in the current directory)

### Structure of a Test Specification directory



* For each test case (aka Transaction, aka Interaction) there exists:
  + A <uniqueName>-req.xml file, this file contains the actual request for the test
  + A <uniqueName>-val.xml file, this file contains the validation of the response, aka the specification of the Minimal Expected Response
  + Optionally a <uniqueName>-rsp.xml file, not used right now, but this file would contain a specific, complete, correct Response which could be used later to automatically generate a virtual service for this API
* Excel files in this directory can be selected via a File Open selection box during a Test Generation run. The excel file will contain a list of test cases that have to be generated. This allows for a granular rebuild of test cases; if only one test case specification is added or amended than not the whole set has to be regenerated.

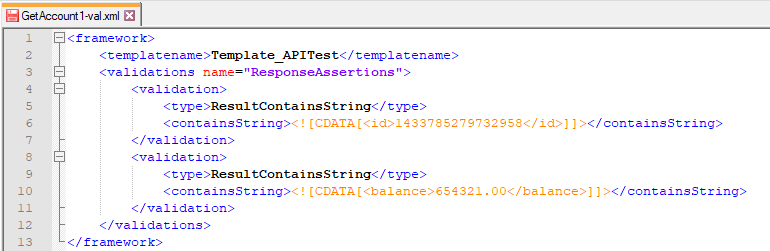
## A Test Specification

### The Request file



* Contains the request payload

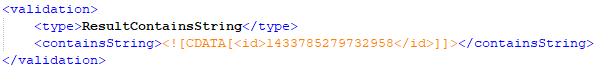
### The Validation file



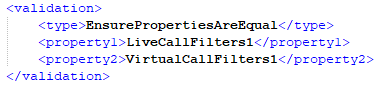
* Contains an xml structure which defines the type of test to be build and all the test specific behavior data to validate the response.
* The structure and content of this xml is dictated by the template that is used to build the required functionality of the tests. Different templates might use more than one “validations” element, and/or use “filters” elements, etc...
* Element “templatename”, must be the name of an existing template in the Tests/Templates directory of the LowCode TestAutomation Generation DevTest project.
* The “name” attribute of the validations and/or filters elements is dictated by the template

### Currently implemented validation types

* ResultContainsString

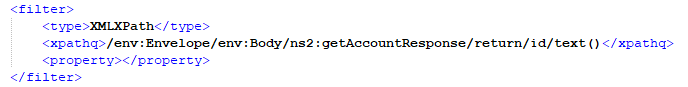


* EnsurePropertiesAreEqual

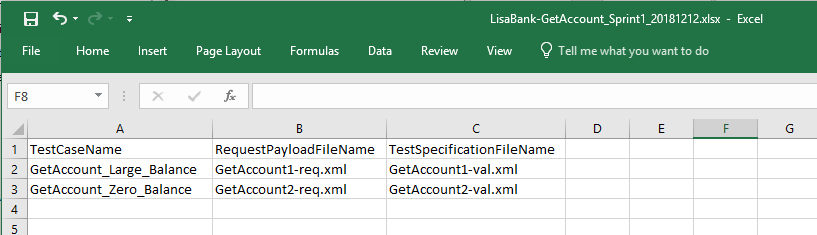


### Currently implemented filter types

* XmlXPath



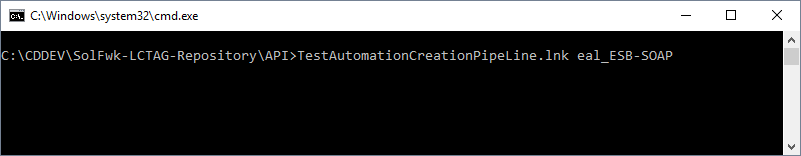
## The Test Suite Specification



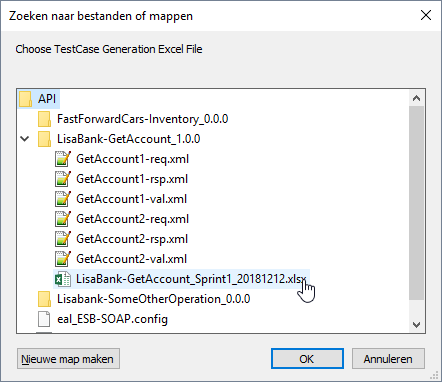
* An excel file that lists the test cases to be build
* Columns
  + TestCaseName, the name of the test case to be generated in the DevTest project
  + RequestPayloadFileName, relative filename of the request payload
  + TestSpecificationFileName, relative filename of the validation specification (the xml “framework” structure)

# A Test Generation Run

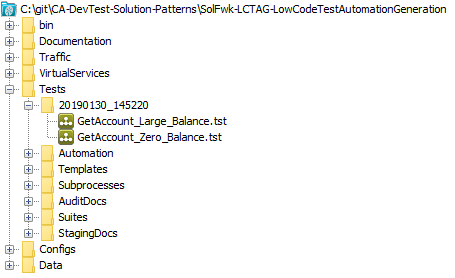
* Start a DOS Command Window
* Navigate to the “Test Specification (Source File) Repository” directory
* Run the command “TestAutomationCreationPipeLine.lnk <configFileNameWithoutSuffix>”



* Select the required Test Suite Specification excel-file



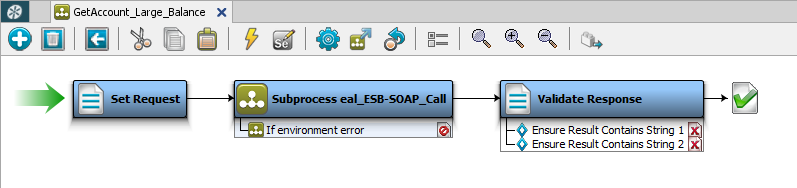
* When the batch file has finished you will find the generated tests inside the LowCode TestAutomation Generation DevTest project within a newly created Tests/<TimeStamp> folder.



# Example Use Cases

## API Tests for operation LisaBank-GetAccount

This use case uses the “Template\_APITest” template. This template creates a simple API test with below flow:



The “Set Request” step defines the payload and stores it in a property. At generation time the request payload originates from the API Repository file defined under the column RequestPayloadFileName in the Test Suite Specification file.

The “Subprocess eal\_ESB-SOAP\_Call” step is a subprocess that does the actual call to the API. This template generates testcases which separates the technical, infrastructure requirements from the functional business requirements of the test. The subprocess takes the request payload as input and exports the response payload as output. The last step contains a set of assertions that will fail the test if the response fails them. At generation time the set of assertions originates from the API Repository file defined under the column TestSpecificationFileName in the Test Suite Specification file.

### Generating the tests

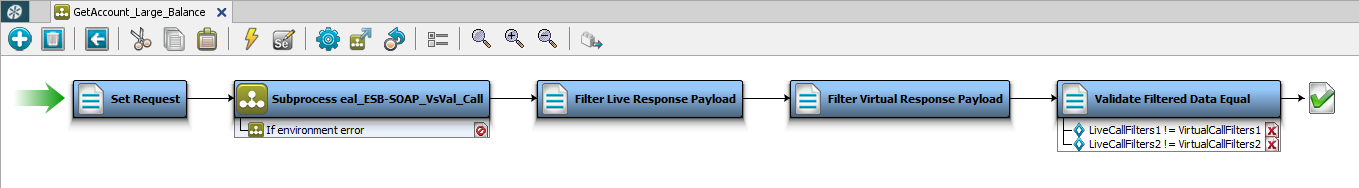
* Start a DOS Command Window
* Navigate to the “SolFwk-LCTAG-Repository\API” directory
* Run the command “TestAutomationCreationPipeLine.lnk eal\_ESB-SOAP”
* In the File Open Dialog select the file “LisaBank-GetAccount\_Sprint1\_20181212.xlsx” in subdirectory “LisaBank-GetAccount\_1.0.0”
* This will generate 2 tests, GetAccount\_Large\_Balance.tst and GetAccount\_Zero\_Balance.tst, in a timestamp-subdirectory

### Testing the generation

* From the SolFwk-LCTAGLowCodeTestAutomationGeneration project, deploy the virtual service “SolFwk-LCTAG\_LisaBank-GetAccount\_Live” using config file “SolFwk-LCTAG\_LisaBank-GetAccount”. This simulates the live API. It will deploy the live simulation of the API on port 8801.
* Run the test cases uses config file “eal\_ESB-SOAP”, both testcases should pass

## Virtual Service Validation for VS LisaBank-GetAccount

This use case uses the “Template\_VSValidation.tst” template. This template creates a test which will validate that the response of a virtual service is in sync with the response of the actual live service. The testcase has below flow:



The “Set Request” step defines the payload and stores it in a property. At generation time the request payload originates from the API Repository file defined under the column RequestPayloadFileName in the Test Suite Specification file.

The “Subprocess eal\_ESB-SOAP\_VsVal\_Call” step is a subprocess that does the actual call to the live API and to the virtual service. This template generates testcases which separates the technical, infrastructure requirements from the functional business requirements of the test. The subprocess takes the request payload as input and exports as output the response payload of the live API as well as the response payload of the virtual service. The “Filter Live Response Payload” step contains a set of filters that select specific content from the live response payload. The “Filter Virtual Response Payload” step contains a set of filters that select specific content from the virtual response payload. The last step contains a set of assertions that will fail the test if the content filtered from the live response is not the same as the content from the virtual response.

At generation time the sets of filters and assertions originates from the API Repository file defined under the column TestSpecificationFileName in the Test Suite Specification file.

### Generating the tests

* Start a DOS Command Window
* Navigate to the “SolFwk-LCTAG-Repository\VirtualServiceValidation” directory
* Run the command “TestAutomationCreationPipeLine.lnk eal\_ESB-SOAP”
* In the File Open Dialog select the file “LisaBank-GetAccount\_Sprint1\_20181212.xlsx” in subdirectory “LisaBank-GetAccount\_1.0.0”
* This will generate 2 tests, GetAccount\_Large\_Balance.tst and GetAccount\_Zero\_Balance.tst, in a timestamp-subdirectory

### Testing the generation

* From the SolFwk-LCTAGLowCodeTestAutomationGeneration project, deploy the virtual service “SolFwk-LCTAG\_LisaBank-GetAccount\_Live” using config file “SolFwk-LCTAG\_LisaBank-GetAccount”. This simulates the live API. It will deploy the live simulation of the API on port 8801. And deploy the virtual service “SolFwk-LCTAG\_LisaBank-GetAccount\_InSyncVirtual” using config file “SolFwk-LCTAG\_LisaBank-GetAccount”. This simulates the Virtual API. It will deploy the virtual service of the API on port 8802. This is a virtual service that is in sync with the live API.
* Run the test cases uses config file “eal\_ESB-SOAP\_VsVal”, both testcases should pass
* For a second test, deploy the virtual service “SolFwk-LCTAG\_LisaBank-GetAccount\_OutOfSyncVirtual” using config file “SolFwk-LCTAG\_LisaBank-GetAccount”. This simulates an out of sync Virtual API. It will deploy the virtual service of the API on port 8803.
* Run the test cases uses config file “eal\_ESB-SOAP\_VsVal\_OutOfSync”, testcase “GetAccount\_Zero\_Balance” will fail.

# Framework High Level Design

Template Name is in the test specification file

API Test Suite can contain tests created from different templates

## CreateTestsV3

# Extending the Framework

## Adding and/or Adapting Templates

## Adding Filter Types

## Adding Assertion Types