SOP for Applet Testing Procedure

Name:	Date:	Prepared By:	Approved By:
SOP for Applet Testing Procedure v1.1	01-06-2022	Technical Support Service	Technical Manager

Ajitav Mohanty

Saji Krishnan

Revision History

Issue	Date	Reason for change
1	20-Dec-2020	First issue
1.1	01-06-2022	Updated and accommodated minor text changes
1.2	03-Jun-2022	Inclusion of Integration testing despite of standalone application testing

Table of contents

Introduction

This document defines the standard procedure for the Applet testing and validation.

The software testing methodology defines as the practice of investigating a software / system under test to ensure that it is of the highest quality.

The three main goals of Software Testing are:

- **Defect Detection:** Find defects / bugs in the software during all stages of its development (earlier, the better).
- **Defect Prevention:** Because of defect detection, help anticipate and prevent defects from occurring at later stages of development or from recurring in the future.
- **User Satisfaction:** Ensure customers / users are satisfied that their requirements (explicit or implicit) are met.

Requirement Analysis Phase – Inputs, Efforts

Requirement Analysis is the first phase of the Applet Testing, and it starts as soon as the Applet Requirement Specification is shared with the testing team. The following points needs to be considered in the process.

- The entry criteria of this phase are the provision of SRS (Software Requirement Specification). It is also recommended that the application architecture is available.
- In this phase, the QA team analyzes at a higher level what to test and how to test.
- The QA team follows up with various stakeholders like Customer, Test Manager/Lead in case of any queries or clarification is required to understand the requirement.
- Requirements may be functional or non-functional like performance, security, usability, etc. or both functional and non-functional.
    - A standard example of Functional vs Non-Functional requirement are as below:

FUNCTIONAL REQUIREMENTS	NON-FUNCTIONAL REQUIREMENTS
A functional requirement defines a system or its component.	A non-functional requirement defines the quality attribute of a software system.
It specifies **"What should the software system do?"**	It places constraints on **"How should the software system fulfill the functional requirements?"**
Functional requirement is specified by User.	Non-functional requirement is specified by technical peoples e.g., Architect, Technical leaders, and software developers.
It is mandatory.	It is not mandatory.
It is captured in use case.	It is captured as a quality attribute.
Defined at a component level.	Applied to a system (as a whole).
Helps you verify the functionality of the software.	Helps you to verify the performance of the software.
Functional Testing like System, Integration, End to End, API testing, etc. are done.	Non-Functional Testing like Performance, Stress, Usability, Security testing, etc. are done.
Usually easy to define.	Usually more difficult to define.
Example **1)** Authentication of the applet with the server whenever he/she provides passkey. **2)** Multi-IMSI Fallback mode.	**Example** **1)** PKI crypto latency of no greater than 7 seconds. **2)** Process open channel requests during limited connectivity.

- The exit criteria of this phase are to complete the RTM document, automation feasibility report and a list of questions if applicable to be more specific on the requirements.

Test_Checklist_v0.1.xlsx

RTM_Traceability_v0.1.xlsx

Test Planning

A document describing the scope, approach, resources, and schedule of intended test activities. It identifies amongst others test items, the features to be tested, the testing tasks, who will do each task, degree of tester independence, the test environment, the test design techniques and entry and exit criteria to be used, and the rationale for their choice, and any risks requiring contingency planning. It is a record of the test planning process.

- To determine the scope and the risks that needs to be tested and that are NOT to be tested.
- Documenting Test Strategy.
- Making sure that the testing activities have been included.
- Deciding Entry and Exit criteria.
- Evaluating the test estimate.
- Planning when and how to test and deciding how the test results will be evaluated and defining test exit criterion.
- The Test artefacts delivered as part of test execution.
- Defining the management information, including the metrics required and defect resolution and risk issues.
- Ensuring that the test documentation generates repeatable test assets.

Test Planning – Process and Template

- The Test Planning starts after the RTM is ready.
- The Plan is built with Requirement details and the Test case details to be reviewed by the Test lead for the test approval.
- The Test plan also covers the Test coverage and the planned Iteration details.
- The Test planning template reference is attached below.

Test_Plan_v0.1.xlsx

Test case development.

Test case in simple terms refers to a documentation which specifies input, pre-conditions, set of execution steps and expected result. A good test case is the one which is effective at finding defects and covers most of the scenarios/combinations on the system under test. Here is the step-by-step guide on how to test case development.

1. Detailed study of the System under test

    a. Before writing test cases, it is very important to have a detailed knowledge about the system which you are testing. It can be any application, applets, or tools. Try and get as much information as possible through available documentation such as requirement specs, SOW, user guides, etc.

    b. Gather all the possible positive scenarios and the odd cases which might break the system (Destructive testing) such as stress testing, uncommon combinations of inputs etc.

2. Written in simple language

    a. While writing test case, it is highly recommended to write in a simple and understandable language.

    b. It is equally important to write your steps to the point and accurate.

    c. Exact and consistent names for e.g., of forms, or fields under test must be used to avoid ambiguity.

3. Test case template

    a. It looks like:

	A	B	C	D	E	F	G	H	I
1	**Test Case ID**	**Test Case Name**	**Description**	**Pre Conditions**	**Execution Steps**	**Expected Result**	**Actual result**	**Status**	**Comments**
2	TC001	Feature_name OR Requirement number/Name as per SRS or client documents		1. 2.	1. 2. 3. 4.	As per requirements			
3									
4									

    b. The Workz Javacard test development is attached as below.

Test_Case_v0.1.xlsx

Test execution environment

A testing environment is a setup of software and hardware for the testing teams to execute test cases. In other words, it supports test execution with hardware, software and network configured.

The Test execution is done using a real Card and simulated Network environment to test the reliability of the product.

In Workz we use different types of Tools which are classified as below:

Test Tools	Test Category
Wojtek	Applet Unit Testing/ Integration Testing
MyTester	Applet Unit test tool/ Functionality testing
PIN PUK Testing	Functionality (Integration) Testing
Milenage Testing	Functionality (Integration) Testing
RFM Tester	Functionality (Integration) Testing
Java Smartcardio Script testing	Unit/Integration/Regression testing /Debugging
Handset Testing (Internal)	Functionality (Integration) Testing
Client UAT	Alpha Testing

Test execution and recording

The developer releases the binary using a timestamp identification and pushing it via GIT. Additionally, a copy is also made in the T-Drive->SIM_Data->Customer repository.

The validator picks the released binary from GIT/T-Drive Customer repository based on the identified timestamp and executes the test cases which is recorded in the test execution document.

Each test is automated based on various in-house tools available. The validator mostly uses the **Workz Java Smartcardio library** to cover most generic test scenarios.

The released binary from the developer to the validator is recorded in the test execution document and the issues are then reported to the developer.

For backward traceability, the applet can always be verified from the test execution document and the cap file with latest timestamp from the GIT.

Test Logs

The Workz card test tools are built to generate readable APDU logs.

Sample of Test log has been attached for reference.

FallbackCheckWithLUNOK1_SIM.log

Test Report

After the validation is successfully completed, a final test report is generated based by the Developer/Validator with final approval via Hellosign.

The final test report will be part of the Applet deliverables to the Profile integrator.

Applet_Test_Report v1.0.docx

Change Request (CR) Management

A change request is a proposal to alter a product or system, often brought up by the client or another team member. During a project, this can happen when a client wants to change or alter the agreed upon deliverables.

A change request should be taken through the following steps:

- Change request initiation and Control
    - Request for changes should be standardized and subject to management review
    - Change requestor should be kept informed
- Impact Assessment
    - Make sure that all requests for change are assessed in a structured way for analyzing possible impacts
- Control and Documentation of Changes
    - A change log should be maintained that tells the date, person details who made changes and changes implemented
    - Only authorized individual should be able to make changes
    - A process for rolling back to the previous version should be identified
- Documentation and Procedures
    - Whenever Applet/tools changes are implemented the procedures and associated document should update accordingly
- Testing and User signoff
    - Software should be thoroughly tested (regression)
- Version Control
    - Control should be placed on the latest Applet source code to make sure that only the latest version is updated. Tag or baseline the latest applet.

Change_RTM_Traceability_v0.1.xlsx

HELLOSIGN

Audit trail

TITLE	SOP_Applet_Testing_v1.1
FILE NAME	SOP_Applet_Testing_v1.1.pdf
DOCUMENT ID	e863f2da14a4dbf23bd8d229cfa65a7b3bbc4e78
AUDIT TRAIL DATE FORMAT	DD / MM / YYYY
STATUS	● Signed

Document history

SENT	**14 / 06 / 2022** 16:47:05 UTC+4	Sent for signature to Saji Krishnan (saji.krishnan@workz.com) and Ajitav Mohanty (ajitav.mohanty@workz.com) from tss@workz.com IP: 94.200.229.6
VIEWED	**14 / 06 / 2022** 16:47:22 UTC+4	Viewed by Saji Krishnan (saji.krishnan@workz.com) IP: 94.200.229.6
SIGNED	**14 / 06 / 2022** 16:47:47 UTC+4	Signed by Saji Krishnan (saji.krishnan@workz.com) IP: 94.200.229.6
VIEWED	**14 / 06 / 2022** 16:49:55 UTC+4	Viewed by Ajitav Mohanty (ajitav.mohanty@workz.com) IP: 94.200.229.6
SIGNED	**14 / 06 / 2022** 16:50:05 UTC+4	Signed by Ajitav Mohanty (ajitav.mohanty@workz.com) IP: 94.200.229.6
COMPLETED	**14 / 06 / 2022** 16:50:05 UTC+4	The document has been completed.