**Functional Test Report for ARC v0.10.0**

**07-06-2024**

# 

[Testing Scope 3](#_Toc168957557)

[Test Approach 3](#_Toc168957558)

[Verified configuration 4](#_Toc168957559)

[Feature Health 4](#_Toc168957560)

[Test execution statistics 5](#_Toc168957561)

[Functional test results by modules 5](#_Toc168957562)

[Detailed Test metrics 5](#_Toc168957563)

# 

# Testing Scope

The scope of testing is to verify fitment to the specification from the perspective of

* Functionality
* Deployability
* Configurability
* Customizability

Verification is performed not only from the end user perspective but also from the System Integrator (SI) point of view. Hence Configurability and Extensibility of the software is also assessed. This ensures the readiness of software for use in multiple countries.

The ARC (Android Reg-Cline) testing scope revolves around the following flows:

* Login into ARC
* Adding machine details
* Consent page
* Demographic detail’s
* Document upload
* Biometric data
* Preview screen
* Authentication screen
* Acknowledgement screen
* Multilanguage support
* Sync/upload/Auto upload packet
* Operator Onboarding

# Test Approach

Persona based approach has been adopted to perform the IV&V, by simulating test scenarios that resemble a real-time implementation.

A Persona is a fictional character/user profile created to represent a user type that might use a product/or a service in a similar way. Persona based testing is a software testing technique that puts software testers in the customer's shoes, assesses their needs from the software and thereby determines use cases/scenarios that the customers will execute. The persona needs may be addressed through any of the following.

* Functionality
* Deployability
* Configurability
* Customizability

The verification methods may differ based on how the need was addressed.

# Verified configuration

Verification is performed on various configurations as mentioned below

* Default configuration - with 3 Lang (Eng, Fra and Ara)

# Feature Health

# Test execution statistics

## Functional test results by modules

Below are the test metrics by performing functional testing using mock SBI and mock ABIS. The process followed was black box testing which based its test cases on the specifications of the software component under test. The functional test was performed in combination of individual module testing as well as integration testing. Test data were prepared in line with the user stories. Expected results were monitored by examining the user interface. The coverage includes GUI testing, System testing, End-To-End flows across multiple languages and configurations. The testing cycle included simulation of multiple identity schema and respective UI schema configurations.

|  |  |  |  |
| --- | --- | --- | --- |
| **Total** | **Passed** | **Failed** | **Skipped (N/A)** |
| 453 | 322 | 67 | 64 |
| Test Rate: 85% With Pass Rate: 82% | | | |

**UI Automation Reports:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Total** | **Passed** | **Failed** | **Skipped (N/A)** |
| 8 | 7 | 1 | 0 |

**Hash Tag:** ca4c66bce6767ce995c78893a61a34fd154f7de1

## Detailed Test metrics

Below are the detailed test metrics by performing manual/automation testing. The project metrics are derived from Defect density, Test coverage, Test execution coverage, test tracking and efficiency.

The various metrics that assist in test tracking and efficiency are as follows:

* Passed Test Cases Coverage: It measures the percentage of passed test cases. (Number of passed tests / Total number of tests executed) x 100
* Failed Test Case Coverage: It measures the percentage of all failed test cases. (Number of failed tests / Total number of test cases executed) x 100

Git hub link for the xls file:

<TBD>