**Functional Test Report for INJI Verify v0.14.0**

**15-09-2025**

# 

[Testing Scope 3](#_Toc209109210)

[Testing results: 5](#_Toc209109211)

[Test Approach 7](#_Toc209109212)

[Verified configuration 8](#_Toc209109213)

[Feature Health 8](#_Toc209109214)

[Test execution statistics 9](#_Toc209109215)

[Functional test results 9](#_Toc209109216)

[UI Automation results 9](#_Toc209109217)

[Verify API Test Rig Automation results 10](#_Toc209109218)

[Detailed Test metrics 10](#_Toc209109219)

# 

# 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 readiness of software for use in multiple countries. Since MOSIP is an “API First” product platform.

Testing scope has been focused on the below features:

* Inji Verify Home page
* Verify Scan Feature
* Verify Upload Feature
* OVP Flow
* Same device flow

Below are the combinations that QA verified and Certified INJI Verify:

**Upload feature Verification:**

1. Windows using Edge, Firefox and Chrome browsers.
2. Android using Edge, Firefox and Chrome browsers.
3. iPhone using Safari, Edge, Firefox and Chrome browsers.
4. MAC using Safari Edge, Firefox and Chrome browsers.

**Scan functionality Verification:**

1. MAC (Laptop) with a front camera of 2 megapixel using Chrome, edge, Firefox and Safari browsers.
2. Windows laptop with a front camera of 2 megapixel using Chrome, edge, and Firefox browsers.
3. Mobile Android phone Tablet with back camera 16 megapixel using browsers Chrome, Edge and Firefox browsers.
4. iPhone and iPad with 12 megapixels back camera Chrome, edge, Firefox and Safari browsers.
5. Verified in low light to scan the QR code
6. Verified scan with blur, cracked, low quality QR codes was verified

**OVP** **functionality Verification with current INJI Verify version:**

1. Windows using Edge, Firefox and Chrome browsers.
2. Android using Edge, Firefox and Chrome browsers, 0.19.0 INJI-mobile.
3. iPhone using Safari, Edge, Firefox and Chrome browsers, 0.19.0 INJI-mobile.
4. MAC using Safari Edge, Firefox and Chrome browsers.

**Same device flow Verification with current INJI Verify version:**

1. Android using Edge, Firefox and Chrome browsers, 0.19.0 INJI-mobile.
2. iPhone using Safari, Edge, Firefox and Chrome browsers, 0.19.0 INJI-mobile.

## Testing results:

Below are the results for Upload, Scan and OVP flow functionality with Windows, Android phone, MAC, Android Tablet, iPad and iPhone with different browsers:





A screenshot of a computer

AI-generated content may be incorrect.

Note : OVP flow supports **0.19.0 inji-mobile build only**

# 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 1 Lang
  + English

# Feature Health

A graph with blue and orange bars

AI-generated content may be incorrect.

# Test execution statistics

## Functional test results

Below are the test metrics by performing functional testing. The process followed was black box testing which based its test cases on the specifications of the software component under test. 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 configurations. The testing cycle included simulation of multiple identity schema and respective UI schema configurations.

|  |  |  |  |
| --- | --- | --- | --- |
| **Total** | **Passed** | **Failed** | **Skipped** |
| 655 | 562 | 93 | 0 |
| Test Rate: 100%, With Pass Rate: 85% | | | |

## UI Automation results

Below section provides details on UI Automation by executing MOSIP functional automation Framework.

|  |  |  |  |
| --- | --- | --- | --- |
| **Total** | **Passed** | **Failed** | **Skipped** |
| 17 | 17 | 0 | 0 |
| Test Rate: 100% With Pass Rate: 100% | | | |

Functional and test rig code base branch which is used for the above metrics is:

Hash Tag: sha256:52554ad1062b28e67973e422f046bbc4b49ddad525498017996ed100422d1915

## Verify API Test Rig Automation results

Below section provides details on UI Automation by executing MOSIP functional automation Framework.

|  |  |  |  |
| --- | --- | --- | --- |
| **Total** | **Passed** | **Failed** | **Skipped** |
| 29 | 29 | 0 | 0 |
| Test Rate: 100% With Pass Rate: 100% | | | |

Functional and test rig code base branch which is used for the above metrics is:

Hash Tag: sha256:7c6529869cec006ae825ed85cc9af57818db11f3ed738de05e71bc9f31dfd532

## 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 the failed test cases. (Number of failed tests / Total number of test cases executed) x 100

Git hub link for the xls file:

<TBD>