Software Test Document

1. Introduction
    1.1. Document Identifier
        1.1.1. Revision A, completed September 29
        1.1.2. Authored by
            1.1.2.1. Peter Banis
            1.1.2.2. Michael Kolar
            1.1.2.3. Klaus Cipi
            1.1.2.4. Robert Olsen
    1.2. Scope
    This document is written to define all necessary tests to ensure the success of our project goals laid out in the SRS. This document is to ensure a unified approach to testing the project components and to that end lays out further requirements, as applicable, on how a requirement is to be tested beyond execution of described unit tests.
    1.3. References
        1.3.1. IEEE
        1.3.2. Software Requirements Specification for our project
        1.3.3. Software Design Document for our project
    1.4. System Overview and Key Features
        1.4.1. See Software Requirements Specification and Software Design Document for this information.
    1.5. Test Overview
        1.5.1. Organization
        There is no distinction between development, testing, quality assurance and configuration management. Hence, there is no line of communication that needs to be specified. For any issues raised by testing tasks, they shall be resolved firstly among those in dispute by another team member. If the issue still exists, they shall be resolved by contacting Dr. Silaghi for his opinion.
        1.5.2. Master test schedule
        Each project milestone defines a set of functionality to be implemented. Testing for that functionality shall occur during that milestone. In some cases, future milestones may alter already tested functions. In those cases, tests shall be repeated to ensure no regression of behavior.
        1.5.3. Resources summary
        Tests require a variety of hardware. This hardware distribution has already been allocated to the team inside the Project Plan, and so shall not be discussed further here. All team members shall test using this provided hardware and, where mentioned in the test case itself, be reviewed thoroughly by another team member. This places requirements only on having a computer and the ability to read and execute the software.

1.5.4. Responsibilities
Testing for the Installation verification tool shall be directed by Peter Banis. Testing for API functionality(Ad-Hoc) shall be directed by Peter Banis and Klaus Cipi. Testing for P2P functionality shall be directed by Robert Olsen and Michael Kolar. All team members are expected to assist in testing, and the one directing testing will be responsible for signing off on each test result.

1.5.5. Tools, Techniques, methods and metrics
The testing process shall make no requirement on the environment being tested except that it meets an Operating System requirement and is capable of executing the test. The nature of the API is such that it will be used on generic devices, and so we shall not test on a non-representative system.

2. Details of the Master Test Plan

2.1. Test processes including definition of test levels

2.1.1. API functionality

2.1.1.1. Test Items and their identifiers

2.1.1.1.1. Ad-Hoc capabilities
This project shall consider the existing API functionality before our work on it as PASS. Any changes to the Ad-Hoc capabilities made to the API, either directly or indirectly, shall ensure the functionality conforms to the existing API functionality.

2.1.1.1.2. P2P capabilities
This project shall confirm adherence to the P2P standard and all necessary test criteria provided by the standard. This section of capability is relevant only when the program using the API has specified to enable P2P capability.

2.1.1.1.3. Communication bridge between Ad-Hoc and P2P
This project shall enable, as much as possible, communication across Androids P2P standard and the existing Ad-Hoc standard. The extent of communication between these two standards is not currently known, and all tests shall be orientated towards reaching and understanding that limit of communication.

2.1.1.2. Features to be tested

2.1.1.2.1. Ad-Hoc capabilities
All existing functionality provided by the API shall be tested for correctness.

2.1.1.2.2. P2P capabilities
All capabilities defined by the P2P standard shall be tested for support. This includes but is not limited to

all requirements for initiating connections, maintaining connections, sending data and receiving data.

2.1.1.2.3. Communication bridge between Ad-Hoc and P2P

The extent to which this is possible has not yet been determined, and so the testing for this functionality shall be testing all the Ad-Hoc functionality when connecting an Ad-Hoc device to an Android device.

2.1.1.3. Features not to be tested

There are no features that will not be tested.

2.1.1.4. Approach

2.1.1.4.1. Ad-Hoc capabilities

As network programming has nuances which are not easily covered by executing code segments in unit tests, all functionality shall be confirmed by both the execution of unit tests, as well as formal review by at least one other team member.

2.1.1.4.2. P2P capabilities

As network programming has nuances which are not easily covered by executing code segments in unit tests, all functionality shall be confirmed by both the execution of unit tests, as well as formal review by at least one other team member.

2.1.1.4.3. Communication bridge between Ad-Hoc and P2P

As this is a primary goal of the project, we shall test each item thoroughly through unit tests as well as review by at least one other team member. Each test must be performed on already working configurations. That is, all tests conducted must be on hardware already confirmed to work in its respective realm(The Android device must already have passed testing on it, and the Linux/Windows/Mac device must already have passed testing on it). This is to reduce if not eliminate the possibility of errors from Ad-Hoc capabilities or P2P capabilities from corrupting the results of the communication bridge.

2.1.1.5. Item Pass/Fail Criteria

2.1.1.5.1. Pass

A test item shall be deemed to pass if and only if it has successfully achieved the intended output for each input, and the code corresponding to that item has been through a proper review procedure, defined 2.1.1.4.

2.1.1.5.2. Fail

A test item shall be deemed to fail if it has either not achieved the intended output for each input, or if the code corresponding to that item has been rejected at the review procedure defined in 2.1.1.4.

2.1.1.6. Suspension Criteria and Resumption Requirements

As the communication bridge between Ad-Hoc and P2P capability contains many unknowns, if the item fails in execution of code and the reason for failure cannot be determined by a code review, the item shall be suspended. It may be resumed following research into the topic, which may include referencing the standard, or asking Dr. Silaghi as our sponsor.

2.1.1.7. Test deliverables

Each test shall have a table of input, output(expected), output(real) and signoff from the reviewing team member.

2.1.2. Installation verification tool

2.1.2.1. Test Items and their identifiers

2.1.2.1.1. File existence checking

The API comes with a set of Java files and Shell/batch scripts that must be checked for existence prior to use. This tool must confirm the existence of all Java files as well as all essential scripts.

2.1.2.1.2. File permissions checking

The API comes with a set of Java files Shell/batch scripts that must be checked for proper permissions to ensure correct execution. This tool must ensure each essential script has at least read/execute permissions, and each Java file has at least read permissions.

2.1.2.1.3. Android device capability checking

Android devices may not all be capable, for a variety of reasons, of supporting the P2P standard. This tool must enable a way to see if the device does support the P2P standard or not.

2.1.2.2. Features to be tested

All features of the tool must be tested.

2.1.2.3. Features not to be tested

No feature of the tool will not be tested.

2.1.2.4. Approach

2.1.2.4.1. File existence checking

In order to test if all the files needed exist, we shall test the tool on all three operating systems(Windows, Mac and Linux). Each of these operating systems is slightly different in how files are stored, and so the tool must ensure it works across all three.

2.1.2.4.2. File permissions checking

In order to test if all the needed permissions exist, we shall test the tool on all three operating systems(Windows, Mac and Linux). Each of these three operating systems is slightly different in how to access this information, and so the tool must ensure it works across all three.

2.1.2.4.3. Android device capability checking

The tool shall have a capability to determine if the Android device supports the P2P standard. Due to the nature of Android vs. Windows/Mac/Linux, it will not be necessary to check any permissions or file existence.

2.1.2.5. Item Pass/Fail Criteria

2.1.2.5.1. Pass

An item shall pass if it has given expected output on all unit tests.

2.1.2.5.2. Fail

An item shall fail if it does not pass all unit tests.

2.1.2.6. Suspension Criteria and Resumption Requirements

An item shall be suspended if it cannot be made to pass its unit tests and there is mutual agreement between all team members to suspend the item. It may be resumed at any time by any team member.

3. Test Plan Test/Use Cases

3.1. API

3.1.1. Ad-Hoc Capabilities

3.1.1.1. Requirement 2.3.1.1

3.1.1.1.1. Objective

The objective of this test case is to ensure the listed requirement is successfully met.

3.1.1.1.2. Input(s)

3.1.1.1.2.1. Hardware/Software requirements

3.1.1.1.2.1.1. A sample program that initiates and sends data across a connection.

3.1.1.1.2.1.2. A Windows 7 device

3.1.1.1.2.1.3. A Windows 10 device

3.1.1.1.2.1.4. A Mac device

3.1.1.1.2.1.5. A Linux device

3.1.1.1.2.1.6. An Android device

3.1.1.1.2.2. Steps to be performed

3.1.1.1.2.2.1. Initiate the program on a Linux device

3.1.1.1.2.2.2. Connect and transfer data with each of the other devices

3.1.1.1.2.2.2.1. Data shall be tested with at least an array of non-zero integers

3.1.1.1.3. Outcome(s)

- 3.1.1.1.3.1. Success
  - 3.1.1.1.3.1.1. Connection established
  - 3.1.1.1.3.1.2. Data transferred with full integrity
- 3.1.1.1.3.2. Failure
  - 3.1.1.1.3.2.1. Case 1
    - 3.1.1.1.3.2.1.1. Connection not established
  - 3.1.1.1.3.2.2. Case 2
    - 3.1.1.1.3.2.2.1. Connection established
    - 3.1.1.1.3.2.2.2. Data transferred without full integrity

3.1.1.2. Requirement 2.3.1.2

3.1.1.2.1. Objective

This test case is to ensure the API is capable of recognizing and relaying to the user of the API the capability of the Android device to operate in either P2P or pseudo Ad-Hoc mode.

3.1.1.2.2. Input(s)

- 3.1.1.2.2.1. Hardware/Software requirements
  - 3.1.1.2.2.1.1. An Android device capable of P2P capability
  - 3.1.1.2.2.1.2. An Android device not capable of P2P capability
- 3.1.1.2.2.2. Steps to be performed
  - 3.1.1.2.2.2.1. Execute installation verification tool “Check P2P support” on a P2P capable Android device
  - 3.1.1.2.2.2.2. Execute installation verification tool “Check P2P support” on a non-P2P capable Android device

3.1.1.2.3. Outcome(s)

- 3.1.1.2.3.1. Success
  - 3.1.1.2.3.1.1. System recognizes the P2P capable Android device as P2P capable
  - 3.1.1.2.3.1.2. System recognizes the non-P2P capable Android device as non-P2P capable
- 3.1.1.2.3.2. Failure
  - 3.1.1.2.3.2.1. The P2P capable Android device is declared non-P2P capable
  - 3.1.1.2.3.2.2. The non-P2P capable Android device is declared P2P capable

3.1.1.3. Requirement 2.3.1.5

3.1.1.3.1. Objective

This test is designed to ensure the ability to distribute information throughout an Ad-Hoc network is maintained. That is, that we are not limited to transferring information between only two devices at a time, which is tested by 3.1.1.1.

3.1.1.3.2. Input(s)

- 3.1.1.3.2.1. Hardware/Software requirements
  - 3.1.1.3.2.1.1. A P2P capable Android device
  - 3.1.1.3.2.1.2. A Mac device
  - 3.1.1.3.2.1.3. A Windows 7 device
  - 3.1.1.3.2.1.4. A Windows 10 device
  - 3.1.1.3.2.1.5. A Linux device
  - 3.1.1.3.2.1.6. A sample program that initiates and sends data across a connection
- 3.1.1.3.2.2. Steps to be performed
  - 3.1.1.3.2.2.1. Connect all devices to an Ad-Hoc network
  - 3.1.1.3.2.2.2. Initiate the program on the Linux device
  - 3.1.1.3.2.2.3. Transfer data throughout the network
    - 3.1.1.3.2.2.3.1. The data shall be at least an array on non-zero integers

3.1.1.3.3. Outcome(s)

- 3.1.1.3.3.1. Success
  - 3.1.1.3.3.1.1. Connections established
  - 3.1.1.3.3.1.2. Data is received with full integrity by all clients in the network
- 3.1.1.3.3.2. Failure
  - 3.1.1.3.3.2.1. Connection established but data not received with full integrity by all clients
  - 3.1.1.3.3.2.2. Connection not established

3.1.2. P2P Capabilities

3.1.2.1. Requirement 2.3.1.4

3.1.2.1.1. Objective

This test is designed to ensure that P2P networks can be constructed and that they do not lose the integrity of any data during transfers.

3.1.2.1.2. Input(s)

- 3.1.2.1.2.1. Hardware/Software requirements
  - 3.1.2.1.2.1.1. Two devices
  - 3.1.2.1.2.1.2. A sample program that initiates and sends data across a connection.
- 3.1.2.1.2.2. Steps to be performed
  - 3.1.2.1.2.2.1. The first device will ping the second device by using its IP address.

3.1.2.1.2.2.2. If the second device responds, then the first device will execute the sample program.

3.1.2.1.2.2.3. The second device shall receive the data via the P2P connection.

3.1.2.1.2.2.3.1. The data shall be at least an array of non-zero integers.

3.1.2.1.3. Outcome(s)

3.1.2.1.3.1. Success

3.1.2.1.3.1.1. The second device will respond to the ping made by the first device.

3.1.2.1.3.1.2. Connection is established.

3.1.2.1.3.1.3. The data is transferred from the first device to the second device with full integrity.

3.1.2.1.3.2. Failure

3.1.2.1.3.2.1. The first device attempts to ping the second device, but does not get a response.

3.1.2.1.3.2.2. Connection is not established.

3.1.2.1.3.2.3. The data received by the second device does not preserve its integrity.

3.1.3. Communication Bridge between Ad-Hoc and P2P

3.1.3.1. Objective

This test case is designed to discover the limits of the communication bridge between Ad-Hoc and P2P capable devices. Hence, there is no strict “success/failure”, only a results table.

3.1.3.2. Input(s)

3.1.3.2.1. Hardware/Software requirements

3.1.3.2.1.1. A P2P capable Android device

3.1.3.2.1.2. An Ad-hoc capable device.

3.1.3.2.2. Steps to be performed

3.1.3.2.2.1. Connect Android device and Ad-Hoc capable device

3.1.3.2.2.2. Go through each function of the Ad-Hoc API, seeing if communication is possible

3.1.3.2.2.3. Record results of each function into a results table

3.1.3.3. Outcome(s)

3.1.3.3.1. Success

3.1.3.3.1.1. Results table created, all Ad-Hoc functions tested and recorded

3.1.3.3.2. Failure

3.1.3.3.2.1. Results table not created

3.1.3.4.

3.2. Installation verification tool

3.2.1. File existence checking

3.2.1.1. Requirements 2.3.2.1, 2.3.2.3

3.2.1.1.1. Objective

This test will ensure the listed requirements are met. It is necessary to ensure the user has all needed files before attempting to use them.

3.2.1.1.2. Input(s)

3.2.1.1.2.1. Hardware/Software requirements

3.2.1.1.2.1.1. API and Installation Verification tool placed into same directory

3.2.1.1.2.2. Steps to be performed

3.2.1.1.2.2.1. Ensure all API files are in directory

3.2.1.1.2.2.2. Start verification tool “Check files” functionality

3.2.1.1.2.2.3. Confirm tool returns “No files missing”

3.2.1.1.2.2.4. Remove all API files from directory

3.2.1.1.2.2.5. Start verification tool “Check files” functionality

3.2.1.1.2.2.6. Confirm tool returns the name of each file missing, and that that set of names includes each missing file

3.2.1.1.2.2.7. Place all but one API file back into the directory

3.2.1.1.2.2.8. Start verification tool “Check files” functionality

3.2.1.1.2.2.9. Confirm tool reports only the missing file as missing

3.2.1.1.3. Outcome(s)

3.2.1.1.3.1. Success

3.2.1.1.3.1.1. Each “Confirm” step in the Steps to be performed has correct output

3.2.1.1.3.2. Failure

3.2.1.1.3.2.1. Any “Confirm” step has incorrect output

3.2.2. File permission checking

3.2.2.1. Requirements 2.3.2.2, 2.3.2.4

3.2.2.1.1. Objective

This test is to ensure the listed requirements are met. It is necessary that the scripts provided with the API have proper permissions, otherwise it could result in a difficult to determine failure.

3.2.2.1.2. Input(s)

3.2.2.1.2.1. Hardware/Software requirements

3.2.2.1.2.1.1. The API and Installation verification tool inside the same directory

3.2.2.1.2.2. Steps to be performed

3.2.2.1.2.2.1. Ensure all scripts are present along with the API

3.2.2.1.2.2.2. Ensure all scripts have at least “Read” and “Execute” permissions

3.2.2.1.2.2.3. Start verification tool “Check permissions” functionality

3.2.2.1.2.2.4. Confirm all scripts are said to have correct permissions

3.2.2.1.2.2.5. Change all scripts to have only “Read” permissions

3.2.2.1.2.2.6. Start verification tool “Check permissions” functionality

3.2.2.1.2.2.7. Confirm all scripts are listed as lacking “Execute” permissions

3.2.2.1.2.2.8. Change all but one script to have “Read” and “Execute” permissions. The remaining script shall have only “Read” permissions.

3.2.2.1.2.2.9. Start verification tool “Check permissions” functionality

3.2.2.1.2.2.10. Confirm only the one script is reported, and it is reported as lacking “Execute” permission

3.2.2.1.3. Outcome(s)

3.2.2.1.3.1. Success

3.2.2.1.3.1.1. All “Confirm” steps in Steps to be executed have correct output

3.2.2.1.3.2. Failure

3.2.2.1.3.2.1. Any “Confirm” step has incorrect output

3.2.3. Android P2P capability checking

3.2.3.1. Objective

3.2.3.2. Input(s)

3.2.3.2.1. Hardware/Software requirements

3.2.3.2.1.1. A P2P capable Android device

3.2.3.2.1.2. A non-P2P capable Android device

3.2.3.2.1.3. Installation verification tool

3.2.3.2.2. Steps to be performed

3.2.3.2.2.1. Run “Check P2P support” functionality on Android device with P2P support

3.2.3.2.2.2. Confirm tool says device is P2P capable

3.2.3.2.2.3. Run “Check P2P support functionality on Android device without P2P support

                        3.2.3.2.2.4. Confirm tool says device is not P2P capable
                3.2.3.3. Outcome(s)
                    3.2.3.3.1. Success
                        3.2.3.3.1.1. All “Confirm” steps in Steps to be executed have correct output
                    3.2.3.3.2. Failure
                        3.2.3.3.2.1. Any “Confirm” step has incorrect output

4. Test Documentation Requirements
    4.1. Each test shall be recorded in a spreadsheet with the following format
        4.1.1. Date tested
        4.1.2. Who tested
        4.1.3. (As applicable, who verified code)
        4.1.4. Sign off from test Director
    4.2. Each test entry shall have optional comments as needed to note any additional necessary information. This may include but is not limited to
        4.2.1. Hardware
        4.2.2. Anomalies during test
        4.2.3. If test results varied between executions of the test
5. Document change procedure and history
    5.1. To change:
        5.1.1. Record information in document identifier(Revision * began [Date])
        5.1.2. Highlight changes
        5.1.3. Have team approve changes
        5.1.4. Submit to Dr. Silaghi for revision approval