Swift Ticket

Test Framework Manual

**Created by**

**Daniel Smullen**

**Jonathan Gillett**

**Rayan Alfaheid**

02/16/2013

Document Revision 1.0

Contents

[Test Framework Organization 3](#_Toc348738584)

[Master Test List 3](#_Toc348738585)

[master.lst 4](#_Toc348738586)

[Test Suites 4](#_Toc348738587)

[Test List File (for Test Suites) 4](#_Toc348738588)

[login.lst 4](#_Toc348738589)

[Test Suites Description File 4](#_Toc348738590)

[logout.dsc 5](#_Toc348738591)

[Test Cases 5](#_Toc348738592)

[Test Case Files 6](#_Toc348738593)

[Temporary Test Case Output Files 6](#_Toc348738594)

[logout\_1.inp 7](#_Toc348738595)

[logout\_1.out 7](#_Toc348738596)

[logout\_1.out.test 7](#_Toc348738597)

[logout\_1.cua 7](#_Toc348738598)

[logout\_1.atf 7](#_Toc348738599)

[logout\_1.dtf 7](#_Toc348738600)

[Test Case Description File 8](#_Toc348738601)

[login\_5.dsc 8](#_Toc348738602)

[Writing Test Cases 9](#_Toc348738603)

[Initial Setup 9](#_Toc348738604)

[Writing Test Cases 9](#_Toc348738605)

[login.inp 10](#_Toc348738606)

[login.out 10](#_Toc348738607)

# Test Framework Organization

All tests are organized into test suites. Test suites represent a collection of related tests that address the testing requirements of an individual component or software feature. They are located inside folders with the same name as the test suite. For example, the */login/* folder contains the login test suite.

Within each test suite are individual test cases. These are the most atomic level of testing and represent each of the individual tests that are executed by the test framework. Each of the test case files has the same name as the test case with differing extensions. For example, the first login suite test case - *login\_1* - would have files starting with *login\_1*, and different extensions.

For each individual test suite and the corresponding test cases, there is a test description file. This contains the title, description, and any dependencies of the test suite or individual test cases. Lastly, both the main test framework and each of the individual test suites contain a test list. The master test list contains the name of each test suite to execute as part of the test framework. In addition, each individual test suite directory (i.e. the */login/* folder) also contains a test list which lists each of the individual test cases to execute. The test lists provide a complete level of granularity in testing at each level as entire suites can be specified as well as individual test cases.

# Master Test List

The master test list contains the list of all the test suites to be executed. It contains the names of each of the individual test suites to execute. Test suites can also be skipped by pre-pending the line with a @ symbol.This allows for control over which test suites to execute, and the ability to skip entire test suites which may be unnecessary for the type of testing (i.e. beta release vs. final release). In addition, comments can be added to the master test list file by starting each line with a # symbol.

The master test list must use the filename *master.lst*and must be located in the root directory of the testing folder. Furthermore, each of the test suites must be subdirectories of the current folder the master test list is located in.

The following is a sample master test list which contains several test suites to execute, as well as the *refund* and *addcredit* test suites which are to be skipped.

|  |
| --- |
| master.lst # The master test list  login  logout  create  delete  sell  buy  @refund  @addcredit |

# Test Suites

Test suites contain several atomic test cases. Each test suite folder also contains a test list file which contains a list of each test case to execute. This provides complete test case granularity; individual test cases can be specified for testing, or to be skipped.

## Test List File (for Test Suites)

Each of the test suites contains a test list file which declares which test cases to execute. Test cases can also be skipped by pre-pending the line with the @ symbol.This allows for control over which tests to execute, and the ability to skip tests which may be unnecessary for the type of testing (i.e. beta release vs. final release). In addition, comments can be added to the test list file by starting each line with a # symbol.The test list file has the same name as the test suite with the extension *.lst****.*** For example, the *login* test suite test list would be named *login.lst*.

The following is a sample test list file for the login test suite containing a list of test cases:

|  |
| --- |
| login.lst login\_1  login\_2  # Test skipped because it is unnecessary  @login\_3  login\_4 |

## Test Suites Description File

Each of the test suites contains a description file which provides the title, test suite description, and dependencies of the test suite. This allows granularity over the test suite dependencies, for example the logout test suite would have a dependency on the login test suite passing. The test suite description file has the same name as the test suite with an extension *.dsc*, for example the login test suite test case list would be named *login.dsc*.

The test suite description file contains the following identifiers which are used to specify the title, description, and dependencies. All of the content from the newline after the identifier to the next identifier or end of file is used when the description file is parsed.

|  |  |
| --- | --- |
| TITLE | Contains the test suite title. |
| DESCRIPTION | Contains the test suite description. |
| DEPENDENCIES | Contains the list of test suite dependencies. |

The following is a sample test suite description file for the logout test suite:

|  |
| --- |
| logout.dsc TITLE  Logout Test Suite  DESCRIPTION  The logout test suite contains test related to the logout  transaction.  DEPENDENCIES  login |

# Test Cases

Test cases are the most atomic level of testing and represent each of the individual tests that are executed by the test framework. Each of the test case files start with the same name as the test suite, followed by an underscore and a number. Thus, each test case has the same name as the test case specified in the test list, except with differing extensions. For example, the first login suite test case - *login\_1* - would have test case files starting with *login\_1* and different file extensions.

## Test Case Files

Each of the test case files are used only for the purpose of testing. These files are used as a reference to verify that the output from the program matches the expected output given the specified test inputs.

The following is a summary of the various test case file extensions, and what input/output data is associated with each file extension.

|  |  |
| --- | --- |
| .inp | Contains the all of the interactive program inputs for the test case that would be entered by a user such as the transaction commands and login credentials. |
| .out | Contains all of the interactive program output such as login prompts and error messages. |
| .cua | The current user accounts file that is used for the purpose of the test, in many cases this file is not modified, but is used to execute the login operation. |
| .atf | The available tickets file which contains a list of available tickets that is read by the program. |
| .dtf | The daily transaction file which contains a list of the daily transactions that occur through interactions with the front-end of the application. |

### Temporary Test Case Output Files

Test case output is produced and written to files with the same filename as the previous files, with *.test* appended. Original test case files are never modified, and are used as exemplars to compare the results of the test with. These exemplar files are read-only. The test files which are written during the execution of the test case have *.test* appended and are used to compare the results of the test case against the expected results in the exemplar files. For example, there may be a sell test case (*sell\_5*) where the daily transaction file is written to, and console output is displayed. This would produce the resultant files *sell\_5.out.test* and *sell\_5.dtf.test.*

The following is an example of the test case files for the sample *logout\_1* test case. This test case performs login transaction and then immediately afterwards performs the logout transaction.

|  |
| --- |
| logout\_1.inp login  Jon  logout |
| logout\_1.out Welcome.  >  Enter username:  Username accepted.  Logout complete. |
| logout\_1.out.test Welcome.  >  Enter username:  Username accepted.  Logout complete. |
| logout\_1.cua Jon\_\_\_\_\_\_\_\_\_\_\_\_\_AA\_000100.00  END\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_000000.00 |
| logout\_1.atf END\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_000\_000.00 |
| logout\_1.dtf 00\_Jon\_\_\_\_\_\_\_\_\_\_\_\_\_AA\_000100.00 |

## 

## Test Case Description File

Test case description files are identical to test suite description files, except they are used to describe individual test cases, not test suites.

Each of the test cases contains a description file. It provides the title, description, and describes dependencies of the test case. This disambiguates test case dependencies. For example, the *login\_5* test case verifies that an error message is displayed if the current user accounts file is corrupted. It has a dependency on the *login\_4* test case, which verifies that the current user accounts file can be opened.

The test case description file has the same name as the test case with an extension *.dsc*. For example, the *login\_1* test case description file would be named*login\_1.dsc*.

The test case description file contains the following identifiers which are used to specify the title, description, and dependencies. All of the content from the newline after the identifier to the next identifier or end of file is used when the description file is parsed.

|  |  |
| --- | --- |
| TITLE | Contains the test suite title. |
| DESCRIPTION | Contains the test suite description. |
| DEPENDENCIES | Contains the list of test suite dependencies. |

The following is a sample test case description file for the *login\_5* test case:

|  |
| --- |
| login\_5.dsc TITLE  Corrupted available tickets file  DESCRIPTION  Repeat the process in login\_4, but read in the invalid available tickets file and verify that an error occurs.  DEPENDENCIES  login\_4 |

# 

# Writing Test Cases

## Initial Setup

First, start by booting up into Linux. A script is available that makes it really easy to rename all of the template files that are used for the test cases instead of having to copy and paste everything manually.

Open up the following file in Linux:

/home/<your name>/.bashrc

And paste the following into the file (if you don’t see it press CTRL + H).

# copy all the template files as the new <name>\_<#> specified

renameall()

{

# Copy files as a new name

find . -type f -name "${1}\*" | grep -Po '(\.\w+)' | xargs -L1 -I {} cp "${1}"{} "../${1}/${1}\_${2}"{}

}

Now that you have this script added, the next step is to get a copy of the source code/repository. Navigate to the directory where you want the project files and do the following:

git clone git@github.com:gnu-user/qa-project.git

## Writing Test Cases

1. First, become familiar with how the test cases are written and organized by reading this manual.
2. Next, open the Test List, Software Requirements Specification, and the project description document.
3. In the Test List, select the test suite you want to write the test cases for. The test suites contain the entire collection of individual test cases.
4. Open the project folder on your computer and navigate to the */tests/templates/* folder. This folder contains all of the template files that you use to create each test case, since many of the files such as the available tickets file (.atf) and current user accounts file (.cua) don’t often change.
5. Create new template files, or copy template files from another test suite in the templates folder and modify the respective files so you can use them as templates for the current test suite.
6. For each test case modify the appropriate files to execute the test case.

For example, the *login\_4* test case you would simply modify the *login.inp* and l*ogin.out* files as follows:

|  |  |
| --- | --- |
| login.inp login  invaliduser | login.out Welcome.  >  Enter username:  Invalid username. |

1. Any time you specify console output in the .out template it is important that you always use the same text output for repeated operations so that they are consistent. Look at the file /etc/strings for a list of output to use, if you add any new output then add it to this file.
2. After changing the template files for the test case in the templates folder, execute the following command in the terminal, while in the template folder.

The following command will copy the template files for the login test case into the */templates/* folder, copy them all to */tests/login/*, and rename them to *login\_4.inp*, *login\_4.out*, *login\_4.atf,* etc.:

$ renameall login 4