Getting Started:
This user manual will describe the Testing Infrastructure provided with JES. It will explain how tests are created and executed and understanding their results.

Creating Tests:

Tests should be created using the Python class unittest. The class unittest provides methods for setting up test cases, running, and cleaning up after execution. Documentation can be found at python unittest documentation.

Test Naming Conventions:

TestExecute relies on the test cases to be named in the following format: 'Test_'. If this convention isn't followed then TestExecute will not be able to find the test case and it will not be added to the test suite.

Test methods should be prefixed with 'test' in order to properly be executed.

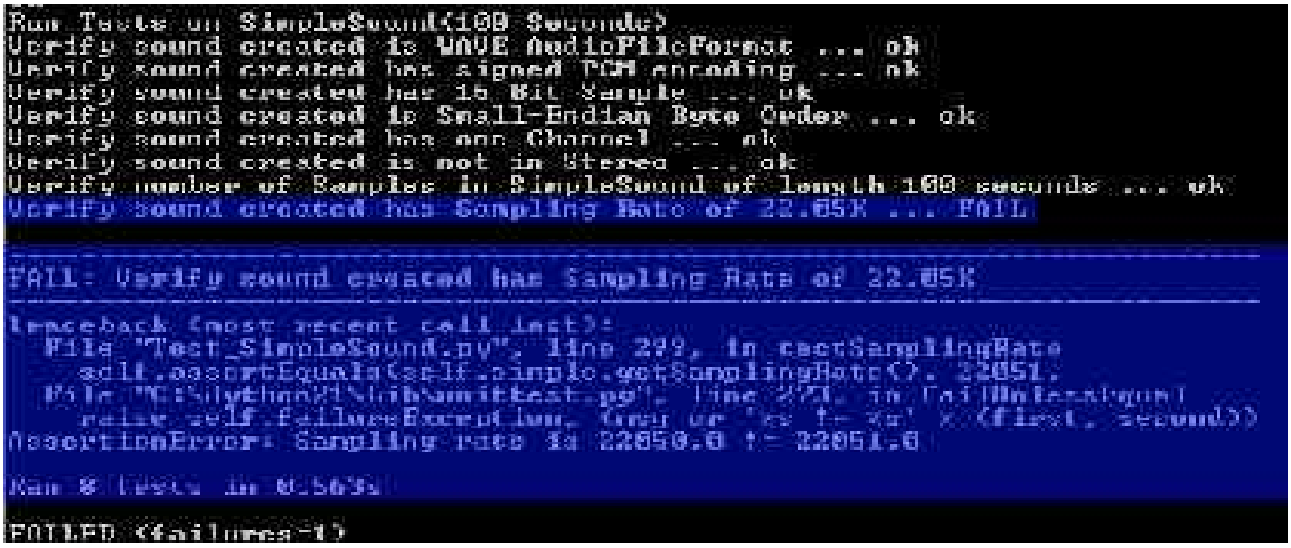
Executing the Tests:

There are two ways to running the test suite. The first way requires that you have Jython installed on your machine. Using Jython to execute the test suite, navigate to the Sources/Tests directory from the command line and simply type 'jython TestExecute.py'.

If you don't have Jython installed you can run the test suite using the shell or batch script found in the Sources/Tests directory. Again navigate to the directory and then if you are using a windows machine type JESTestUtil.bat other wise type JESTestUtil.sh.

Examine the Test Results:

As the errors occur they are printed to the screen. Don't worry about attempting to see what they are as tests continue executing. The failures and errors are reprinted after the last test is run.
(Above shows the results being printed out as the test case is executed)



You'll notice in the picture above the failure is highlighted in blue. You will also see here that the counts are printed out for each individual test case run. Again these can be ignored because the suite compiles complete counts for all test cases at the end of execution.

When the test suite completes the errors are reprinted and the counts are displayed. Shown below the total tests run are highlighted in green, the failures in yellow and the errors in red.

Failures are when the test executes without any problems, but it generates a result that wasn't expected. For example the test could have been expecting to generate the number 16, but instead generated the number 12.

An error means the test case was unable to execute properly. This can occur when a file is missing, such as a picture or sound that is required to run the test. An error can also occur if there is a syntax or programming error.

Understanding a Failure or Error Message:

(Failure message)
In a failure message the very first thing shown is the test methods name. Here shown in purple the test that has failed is 'Verify sound created has Sampling Rate of 22.05K.' Following the test method the trace back is shown. The trace back is composed of the file

where the test method is located, the line number, and the assertion. The assertion is the line of code where the method tests a value to compare to an expected value.

Continuing with the example above the method that failed can be found in 'Test_SimpleSound.py' on line 299 (shown in Teal). The assertion error (shown in red) is due to the fact that it was expecting 22051 but generated 22050 for the Sampling rate.