International Journal of Engineering and Technical Research (IJETR)
ISSN: 2321-0869, Volume-3, Issue-10, October 2015

SCSI/SAS Compliance Testing Tool Using Robot Framework

Sarita M Mantri, Srinivasan Lakshminarayanan, Chidananda Murthy P

Abstract— **All telecom, network companies use Python test frameworks for protocol compliance testing. In the area for storage solutions, SCSI/SAS compliance testing presents unique challenges for framework design. The framework should accommodate deviations in command support, field support, response of the device under test to the command, variations in responses. A prototype for different architectures for device configuration and also a prototype mapping of Python data structures to SCSI/SAS specific formats is proposed. An external SCSI/SAS library for Robot Framework is built to support compliance testing.**

***Index Terms*— SCSI, SAS, Robot Framework, Open Source, QEMU, Emulator, Ctype Module, Python, Test Case**

I. INTRODUCTION

SCSI/SAS Compliance Testing plays a critical role in the area of storage solutions. Using customized framework and scripting languages such as Python, Perl to execute the test cases could be expensive and time consuming. Automated testing is an effective way of test case execution and when implemented with a framework is more effective in improving the test effort and quality. The goal of Automation is to increase the overall productivity and reduces the overall cost of the development. Automated results reports with more accurate interpretation of test execution. Higher efficiency can be achieved using test automation framework which ensures maximum reusability. It is a collection of tools, libraries that provide support for automated compliance testing of SCSI/SAS. Robot Framework is a generic test automation framework for acceptance testing and the acceptance test driven environment. In Robot Framework, Keyword driven scripting technique is used for automated compliance testing that separates the test creation process.

Serial Attached SCSI (SAS) is a data-transfer technology for moving data to and from storage devices. SAS rely upon point-to-point serial protocol that replaces parallel SCSI bus technology that is usually used in servers, data centers, workstations. SCSI/SAS command compliance testing insures that the peripheral under test can properly interpret and execute any valid SCSI/SAS CDB issued to it and also reports the appropriate error information if an invalid SCSI/SAS CDB is issued. SCSI/SAS CDB compliance testing is done using Robot Framework by developing an external test library.

Section II discusses related work, Section III discusses overview of Robot Framework, Section IV discusses proposed system, Section V discusses results and conclusion and further work is presented in section VI. Units

Sarita M Mantri, Computer Science & Engineering, School Of Engineering & Technology, Jain University, Bangalore, India, 8197684060, sarita.mantri@gamil.com.

Srinivasan Lakshminarayanan, Research Scholar, Computer Science & Engineering, School Of Engineering & Technology, Jain University, Bangalore,India, srinirad@gmail.com

Chidananda Murthy P, Asst Professor, Computer Science & Engineering, School Of Engineering & Technology, Jain University, Bangalore ,India, chidananda.murthy@gmail.com

II. RELATED WORK

Lakshman Priyatham Rallapally's paper Robot Automation Framework: A Brief Survey explores on Robot Automation Framework as an efficient test automation framework. It is a Keyword driven scripting testing technique which reduces cost and effort to ensure quality.

Sandeep Sivanandan, Yogeesha C. B's paper Agile Development Cycle: Approach to Design an Effective Model Based Testing with Behaviour Driven Automation Framework explores a design of a Behaviour driven test automation framework using MBT and also its effectiveness when used during Agile Development. The automation framework is analyzed after the unification of the Graphwalker which is a Model Based Graphical User Interface test generator with behaviour driven development framework and Robot Framework.

Tuomas Pajunen, Tommi Takala, and Mika Katara's paper Model-Based Testing with a General Purpose Keyword-Driven Test Automation Framework focuses on the unification of the TEMA model-based graphical user interface test generator with a Robot Framework which is keyword-driven test automation framework. The intention of the unification was to enable the wide testing library support of Robot Framework to be used in online model-based testing.

Liu Jian-Ping, Liu Juan-Juan, Wang Dong-Long's paper Application Analysis of Automated Testing Framework Based on Robot discusses the use of testing tools in software test automation by automation test analysis of Robot Framework.

III. OVERVIEW OF ROBOT FRAMEWORK

Robot Framework is solely developed by Nokia Siemens communication technology limited company. But now it’s open source software and based on the Python language keyword driven automated test framework.Test cases are executed using command line interface. Three output files are generated, HTML based Report and log files in readable format and XML based file. In Robot Framework customized test libraries can be created using simple library API.

Using Robot Framework there is less amount of maintenance work related to testing. It can be extendable using Python, C# & Java. Test cases can be created in uniform way as it allows to use tabular syntax. It ensures easy adaptability and sustainable as it is possible to create new higher-level keywords by grouping and binding existing keywords together. These higher-level keywords are called user keywords to set apart them from lowest level library keywords that are implemented in test libraries.

A. Architecture of Robot Framework

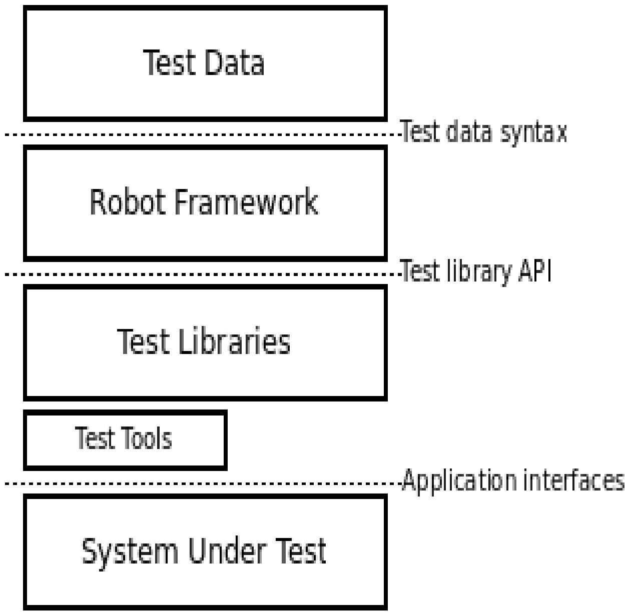


Fig 1 Top Level Architecture of Robot Framework

RF high level architecture is as shown in fig 1 and it consists of four main levels: Test data, RF core, test library/test tools and system under test. Test case and resource files together form the test data. Resource files consist of higher level user keywords, or variables. The Robot Framework engine consist of program written using Python language which will interpret those user keywords written in test data and execute commands in system under test using test library keywords.

The available keywords of Robot Framework are defined in libraries. New libraries can be created using Python or Java programming language. There are two kinds of libraries, standard and external. The standard libraries are distributed with Robot Framework and the external libraries are released in separate packages. There are some supporting tools built-in to Robot Framework such as Libdoc, Rebot, Testdoc and Tidy.

IV. PROPOSED SYSTEM

CDB validation is done before device is manufactured as time to market is very less. CDB compliance testing is done using virtualizer. QEMU is a generic and open source machine emulator and virtualizer. QEMU as a virtualizer performs in an optimal way by executing the guest code directly on the host machine. A virtual SAS disk is created on QEMU and commands are issued to SAS disk using Python utility library.

Python utility library is a small Python package, developed using ctypes module. It sends despotic SCSI/SAS commands to virtual SAS disk through the Linux SCSI Generic driver which provides the SG_IO ioctl for the purpose of communication to SAS disk.
A test suite is executed, which is a collection of written test cases that are intended to be used to test for the validation of CDB of SAS/SCSI using the developed external library for Robot Framework to insure that the peripheral under test can properly interpret and execute any legal SCSI/SAS CDB that it receives and also reports the proper error information if it receives an illegal SCSI /SAS CDB.

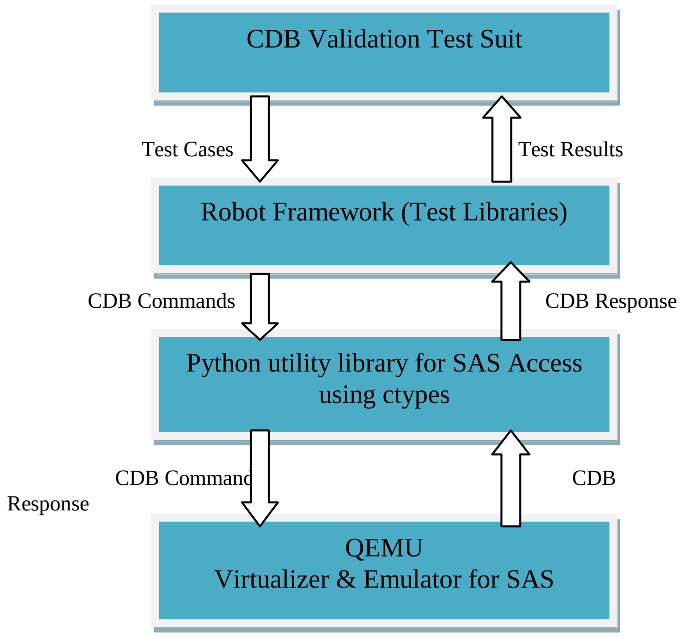

CDB Validation Test Suit
Test Cases
Test Results
Robot Framework (Test Libraries)
CDB Commands
CDB Response
Python utility library for SAS Access using ctypes
CDB Command
CDB
Response
QEMU
Virtualizer & Emulator for SAS

Fig 2 Top Level Architecture

A. Experimental Setup

There are three modules.

Robot Framework Library - Designed test library cdbLibrary.py has static API. Library has a class cdbLibrary with methods which map directly to keyword names. Keywords also take the same arguments as the methods implementing them. Keywords report failures by writing to standard output as exceptions or log and it returns values through the return statement. Libraries implemented as classes can have an internal state. It is important to make sure that changes in one test case do not affect other test cases by chance as the state can affect the behavior of keywords. Robot Framework creates new instances of test libraries for every test case by default. Developed test library scope is TEST SUITE. A new instance is created for every test suite.

Keyword names used in the test data are compared with method names to find the method implementing these keywords. Name comparison is case-insensitive and also spaces and underscores are ignored. The information about the number of required arguments for keywords can be taken directly from the method that implements keywords in a static and hybrid API. After a method implementing a keyword is called, it can use any mechanism to communicate with the system under test. It can also send messages to Robot Framework's log file, returns information which can be saved in variables as well as reports, keyword response namely passed or not. Table I shows the sample test cases. The test cases for SCSI/SAS command is written in text file based on the specification mentioned in reference manual. Table II shows the test case for Inquiry command.

Table I Sample Test Case

Test Item	Test Condition	Test Steps	Expected Result
Inquiry - CDBVal - Normal CDB with Zero ALLOCATION LENGTH	EVPD=0b, PAGECODE= 00h, ALLOCATION LENGTH = 0	Issue Inquiry command with EVPD=0b, PAGECODE =00h, ALLOCATION LENGTH = 0	The Command should be executed with GOOD status.
TestUnitReady-CDBVal-Normal CDB		Issue TestUnitReady command.	Command should be executed successfully.

Table II Robot Framework Test Case

Setting	Value	Value	Value
Library	cdbLibrary.py		

Test Case	Action	Argument	Argument	Argument	Argument	Argument	Argument	Argument
Inquiry - CDBVal - Normal CDB	Inquiry	Opcode =0x12	CMDDT =${0}	EVPD =${0}	RF =${0}	Pagecode =${0}	Allocl en =${96}	control =${0}
	inqr esult should be	Success=${0}						

Python Utility Library - ctypes is a foreign function library for Python. It provides C compatible data types and allows calling functions in DLLs or shared libraries. It can be used to wrap these libraries in pure Python. Shared libraries can be loaded by using one of the prefabricated objects, which are instances of the LibraryLoader class, either by calling the LoadLibrary method, or by retrieving the library as attribute of the loader instance. Foreign functions can be borrowed as attributes of loaded shared libraries. The function objects created by default accept any number of arguments, accept any ctypes data instances as arguments and return the default result type specified by the library loader.

Designed class SgIoHdr, CDB, Response, SenseBuffer. SgIoHdr - The header structure sg_header is a controlling layer between the application and the kernel driver. This structure describes how the execution of SCSI/SAS command is carried out and also stores the results of the execution of the SCSI/SAS command.

CDB - A generic class designed to accept command of different byte size.

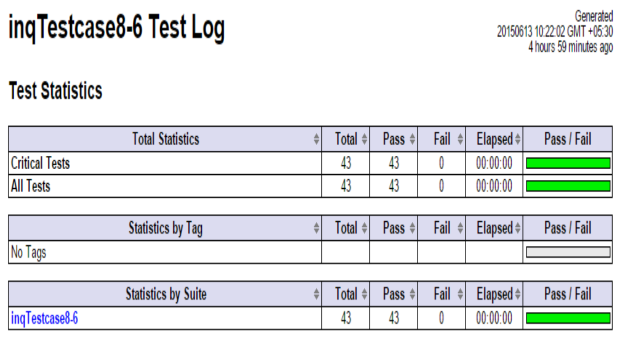
Response - A generic class designed to decode the output data, which is part of the header structure.

SenseBuffer - SCSI/SAS Commands with no output data can give status information through the sense buffer, which is part of the header structure sg_header. If the previous command has terminated with a CHECK CONDITION status then sense data is available in sense buffer. In this case the kernel automatically retrieves the sense data through a REQUEST SENSE command.

QEMU SAS Emulator - QEMU emulates existing hardware and common drivers. It supports multiple operating systems.

V. RESULTS AND DISCUSSION

In Robot Framework logs and reports are generated in a very readable format in HTML files. Test execution on completion will generate three types of files, including log.html, report.html and output.xml. Logs are presented in a tabular format. The background colour is green if the test cases are passed, otherwise it is red. Summary of the test cases executed is given by report file. It includes information such as their names, execution time, duration, and final result. Logs provide more information on the execution, including all the keywords. Logs help in debugging as execution flow can be easily traced using logs. Robot Framework saves the log into an XML file after executing the test cases. Based on the tester's requirement, XML file can be converted into an HTML file or processed further for integration with other possible continuous integration systems.



inqTestcase8-6 Test Log

Generated
20150613 10:22:02 GMT +05:30
4 hours 59 minutes ago

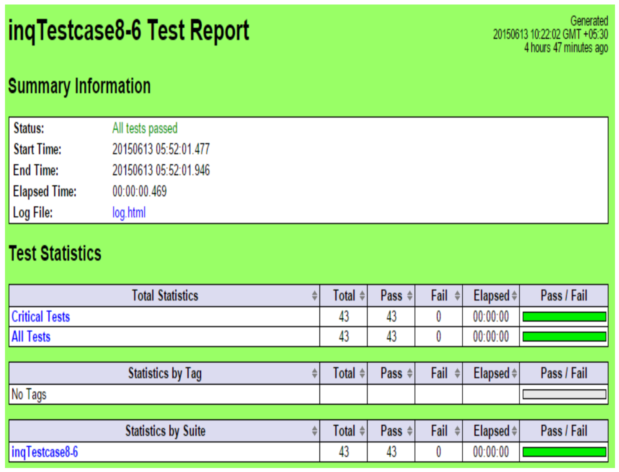
Test Statistics

Total Statistics	Total	Pass	Fail	Elapsed	Pass / Fail
Critical Tests	43	43	0	00:00:00	🟩
All Tests	43	43	0	00:00:00	🟩

Statistics by Tag	Total	Pass	Fail	Elapsed	Pass / Fail
No Tags					

Statistics by Suite	Total	Pass	Fail	Elapsed	Pass / Fail
inqTestcase8-6	43	43	0	00:00:00	🟩

Fig 3 Test case log in Robot Framework



inqTestcase8-6 Test Report

Generated
20150613 10:22:02 GMT +05:30
4 hours 47 minutes ago

Summary Information

Status:	All tests passed
Start Time:	20150613 05:52:01.477
End Time:	20150613 05:52:01.946
Elapsed Time:	00:00:00.469
Log File:	log.html

Test Statistics

Total Statistics	Total	Pass	Fail	Elapsed	Pass / Fail
Critical Tests	43	43	0	00:00:00	🟩
All Tests	43	43	0	00:00:00	🟩

Statistics by Tag	Total	Pass	Fail	Elapsed	Pass / Fail
No Tags					

Statistics by Suite	Total	Pass	Fail	Elapsed	Pass / Fail
inqTestcase8-6	43	43	0	00:00:00	🟩

Fig 4 Test case report in Robot Framework

According to reference manual, on an average 30 test cases are written for 10 SAS/SCSI commands such as Inquiry , Read(6), Write(6), Request Sense, Log Sense, Mode Sense, Test Unit Ready, Log Select, Mode Select, Read Capacity. Fige 3 shows the log file and fig 4 shows the report file generated after successful execution of inquiry command.

VI. CONCLUSION AND FURTHER WORK

Currently there is no open source SCSI/SAS compliance test library. Hence this work can also results in cost reduction. Bringing SCSI/SAS compliance testing to Robot framework will open up looking at using Robot Framework for testing other protocols. Further work can be done on design of Python data structure for CDB, Complex test cases for Robot Framework and design of data driven QEMU SAS emulator which will behave according to the specification.

REFERENCES

[1] http://robotframework.org/
[2] Lakshman Priyatham Rallapally. Masters in Engineering (CS), Research & Development. Engineer Nokia Siemens Networks India Private Ltd. Robot Automation Framework: A Brief Survey. International Journal of Computer Science and Electrical Engineering (IJCSEE) ISSN No. 2315-4209, Vol-1 Iss-1, 2012
[3] Sandeep Sivanandan, Yogeesha C. B. Agile Development Cycle: Approach to Design an Effective Model Based Testing with Behaviour Driven Automation Framework
[4] Tuomas Pajunen, Tommi Takala, and Mika Katara. Model-Based Testing with a General Purpose Keyword-Driven Test Automation Framework.2011 Fourth International Conference on Software Testing, Verification and Validation Workshops.
[5] Liu Jian-Ping, Liu Juan-Juan, Wang Dong-Long. Application Analysis of Automated Testing Framework Based on Robot. 2012 Third International Conference on Networking and Distributed Computing.
[6] SCSI Command Reference Manual, Seagate Corp, December 2010.
[7] http://wiki.qemu.org/Main_Page
[8] https://docs.Python.org/2/library/ctypes.html

AUTHORS

First Author
Sarita M Mantri, MTech (CSE), SET, Jain University, Bangalore
sarita.mantri@gmail.com
Second Author
Srinivasan Lakshminarayanan.
Research Scholar, SET, Jain University, Bangalore
srinirad@gmail.com
Third Author
Chidananda Murthy P.
Asst Professor, CSE Dept, SET, Jain University, Bangalore
chidananda.murthy@gmail.com
Correspondence Author
Sarita M Mantri, sarita.mantri@gmail.com , 8197684060