OIC Conformance Test Tool

USER

MANUAL

TABLE OF CONTENTS

1. GENERAL INFORMATION

1. GENERAL INFORMATION

General information section introduces Conformance Test Tool and purpose for which it is intended.

1.1. Introduction

Open Interconnect Consortium (OIC) Conformance Test Tool (CTT) is a tool used in IoTivity and OIC product to ensure that an IoTivity or OIC product is complaint with OIC standard.

1.2. Organization of the Manual

The user manual consists of four sections: General Information, Prerequisite, Build, and Run.

General Information section explains in general terms the system and the purpose for which it is intended.

Prerequisite section describes installation process of required software and tools that is essential to use Conformance Test Tool.

Build section provides the build process of Conformance Test Tool and DUT Simulator.

Run section explains the execution process of DUT Simulator and Conformance Test Tool using CLI and GUI.

2. PREREQUISITE

2. PREREQUISITE

To install all the prerequisites automatically, go to *iotivity/tools/conformance-test-tool* and run the following command in terminal.

```
$ ./configure.sh
```

If there is any issue, please follow the following procedures (from ***2.2 to 2.7***) to install them manually.

2.1. Platform:

Operating System: Ubuntu 12.04 or higher

Architecture: 32 bit

2.2. Java:

Java 1.8 or higher (To install jdk 1.8, run the following commands)

```
$ sudo add-apt-repository ppa:webupd8team/java
$ sudo apt-get update
$ sudo apt-get install oracle-java8-installer
```

2.3. Gradle:

Gradle 2.3 or higher(To install Gradle on linux, run the following commands)

```
$ sudo add-apt-repository ppa:cwchien/gradle
$ sudo apt-get update
$ sudo apt-get install gradle
```

2.4. Jython:

Jython 2.7 or higher (To install Jython, follow these steps)

- Go to the following Link and Download Jython 2.7.0 installer

  http://www.jython.org/downloads.html
- After download is complete, run the jar file using java-8
- Follow the installation procedure.

2.5. Robot Framework:

To install Robot Framework, follow these steps

- Go to the following link and download zip

  https://github.com/robotframework/robotframework
- Extract the zip file and go to the root folder of robot framework, open terminal and run:

  ```
  $ jython setup.py install
  ```

2.6. Tshark:

To install tshark, follow below commands

```
$ sudo apt-get install tshark
$ sudo chmod 4711 `which dumpcap`
```

2.7. Maven:

- Download Maven (version 3.3.9) from:

  http://www.eu.apache.org/dist/maven/maven-3/3.3.9/binaries/apache-maven-3.3.9-bin.zip

- Ensure ***JAVA_HOME*** environment variable is set and points to your JDK installation
- Extract distribution archive in any directory
- Add the ***bin*** directory of the created directory ***apache-maven-3.3.9*** to the ***PATH*** environment variable
- Confirm with ***mvn -v*** in a new shell.
- For more information about **Maven** installation, go to the link:

  https://maven.apache.org/install.html

2.8. Asynchronous Event Notification Library:

Asynchronous Event Notification Library is needed when building DTLS Relay. To install it, follow the bellow command:

```
$ sudo apt-get install libevent-dev
```

3. BUILD

3. BUILD

3.1. IoTivity Build (For Conformance Simulator)

- Go to the top (root) directory of 'IoTivity' project.
- Follow the prerequisite steps described for IoTivity project.
- Run the below command:

  To Build without Security:

  ```
  $ scons TARGET_OS=linux
  ```

  To Build with Security:

  ```
  $ scons TARGET_OS=linux SECURED=1
  ```

  (Note: C sdk requires tinycbor. Please follow the instruction in the build message to install tinycbor).

3.2. CTT Library, Conformance Simulator and GUI Build

- Go to the root folder (*iotivity/tools/conformance-test-tool*)
- From the terminal, run:

  ```
  $ sudo chmod 777 build.sh
  $ ./build.sh
  ```

3.3. Conformance Simulator (Standalone) Build

Conformance Simulator is an OIC client-server simulator which supports the basic client and server features mentioned in the OIC Core Spec. It is based upon IoTivity implementation of OIC specification. The server part supports discoverable, non-discoverable, observable, non-observable, secured and non-secured resources. Both the server and client support all CRUDN operations. Resource, Platform, Device and Collection discovery is supported for the client. The client supports both CON & NON type messaging.

If only Conformance Simulator build is required, then follow the bellow steps:

- Perform steps described in 3.1
- Go to the top (root) directory of 'ConformanceSimulator' project (conformance-test-tool/res/ConformanceSimulator).
- Run the below command to build Conformance Simulator

  ```
  $ scons
  ```

4. RUN

4. RUN

4.1. Prerequisite

- Create a description of device/resource representation data structure in json format. An example is provided in the conformance-test-tool/json directory. Name the file as 'DUTDescriptor.json'
- Copy the file to *<iotivity_root>/tools/conformance-test-tool/bin/linux/ConformanceTestTool/libs/* folder. (When using the Conformance Simulator as DUT, no need to make any DUTDescriptor.json)
- If DUT device is not available, then run Conformance Simulator by following section-4.4

4.2. Using GUI

a) Start the DUT device or Conformance Simulator (*<iotivity_root>/tools/conformance-test-tool*/bin/linux/ConformanceSimulator)

b) Go to the root folder of Conformance Test Tool, Then Run:

```
$ cd bin/linux/ConformanceTestTool
```

c) To run the GUI App, execute the following:

```
$ ./CTT
```

d) The GUI will start like this:

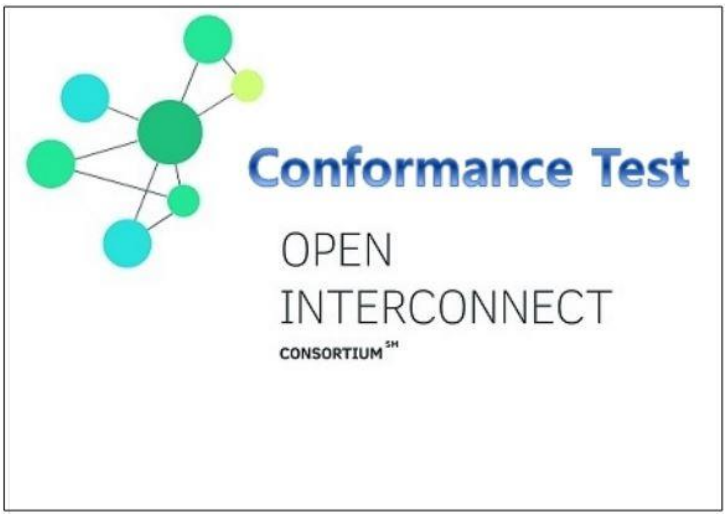

Figure 4.2.1: Splash Screen

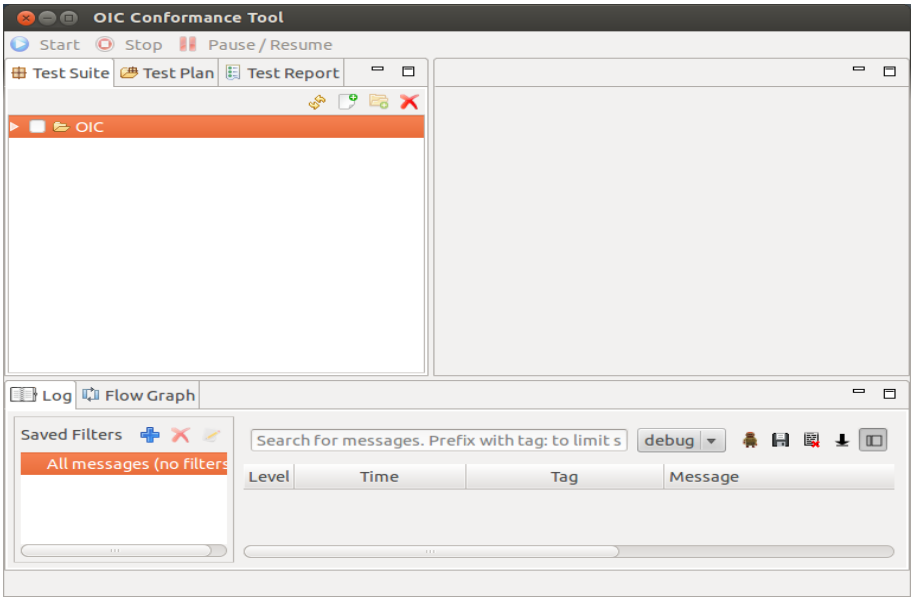

Figure 4.2.2: Conformance Test Tool Graphical User Interface (GUI)

e) How to Create a Test Plan

- Select your desired test cases in the **Test Suite** tab. Click Create Test Plan

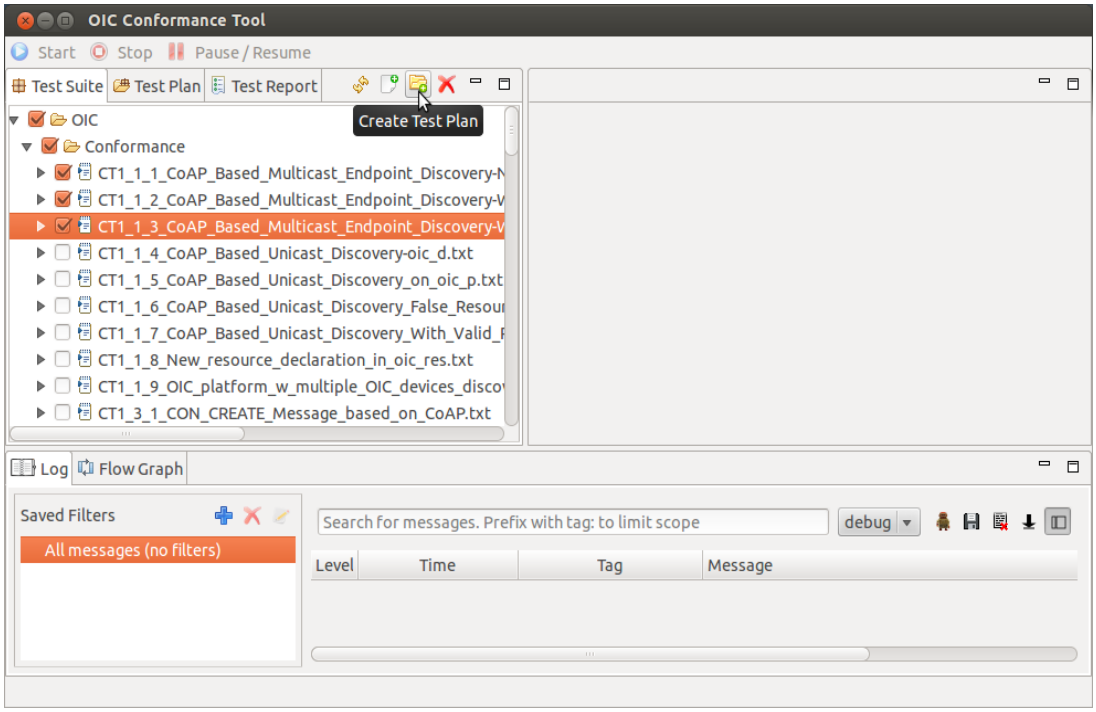

Figure 4.2.3(a): Conformance Test Tool Create Test Plan

- A **Create Test Plan** dialogue box will appear. Enter a name for it and click OK.

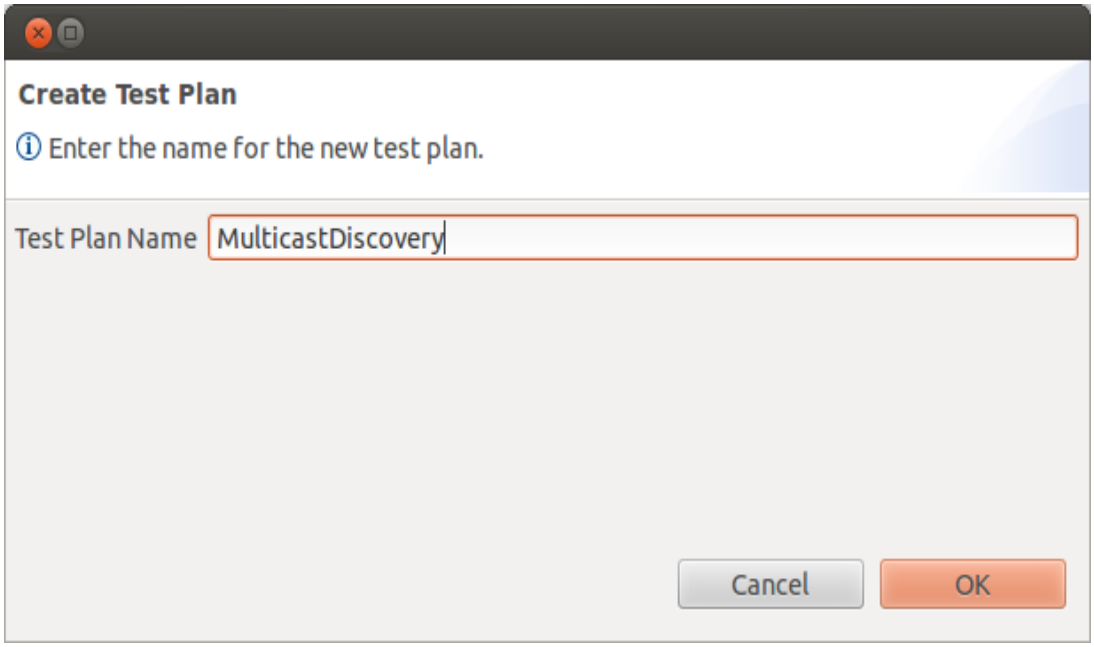

Figure 4.2.3(b): Conformance Test Tool Create Test Plan

- A test plan will be created. The created test plan will be visible in the **Test Plan** tab.

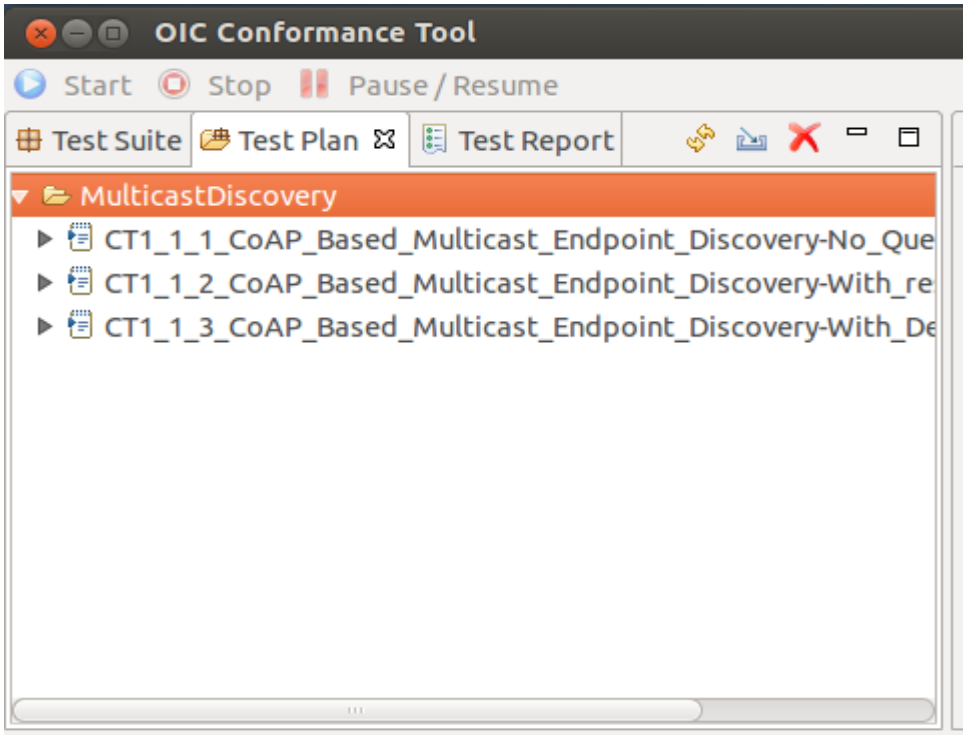

Figure 4.2.4: Conformance Test Tool Show Test Plan

- Double click on the test plan to select it.

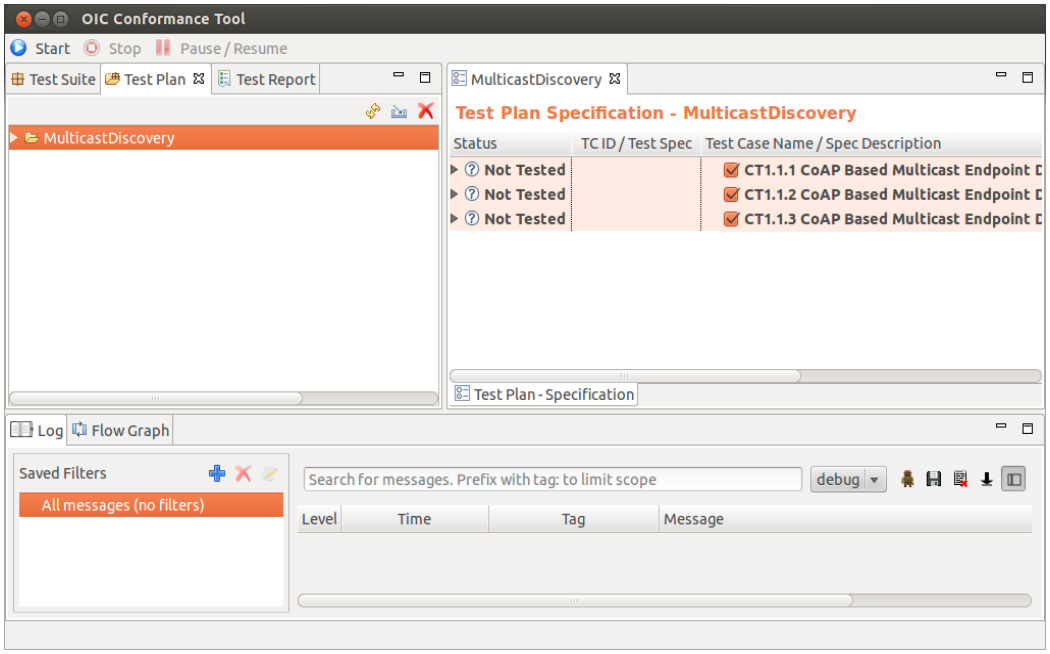

Figure 4.2.5: Conformance Test Tool Select Test Plan

f) Create and Update *DUTDescriptor.json* File

- Go to the *RAMLs* View by clicking on *RAMLS* Tab. It shows all RAML file(s).

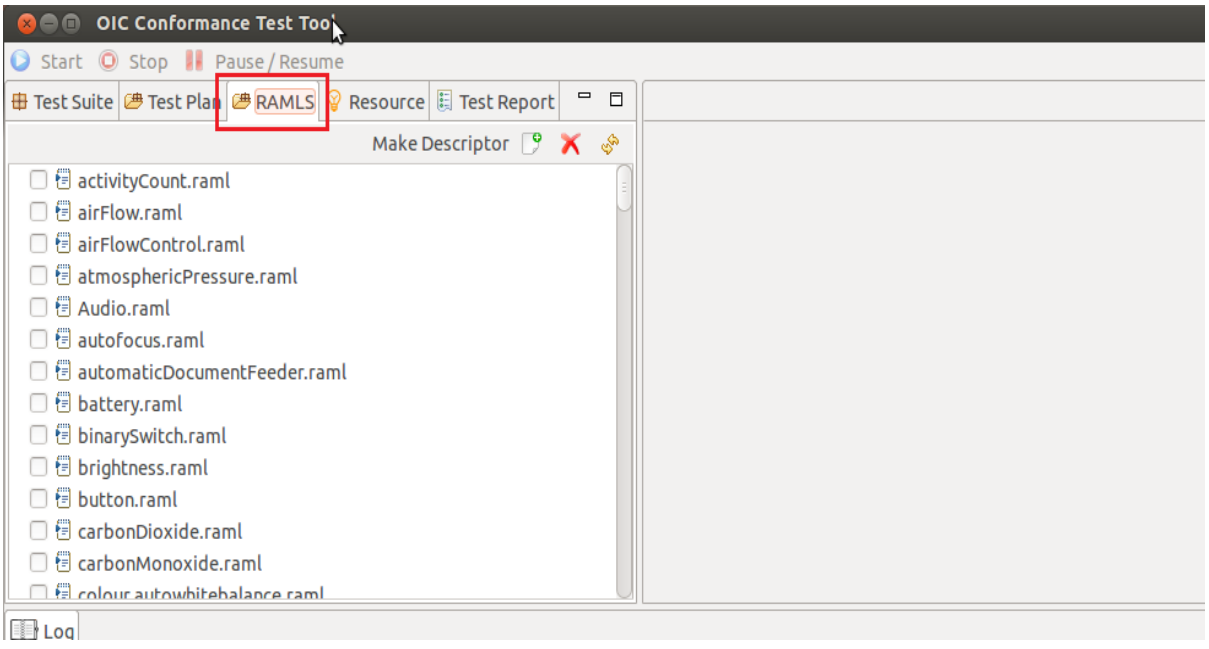

Figure 4.2.6: Conformance Test Tool RAMLs file list

- You can import RAML file(s) from your local storage. Click on *Import* icon.

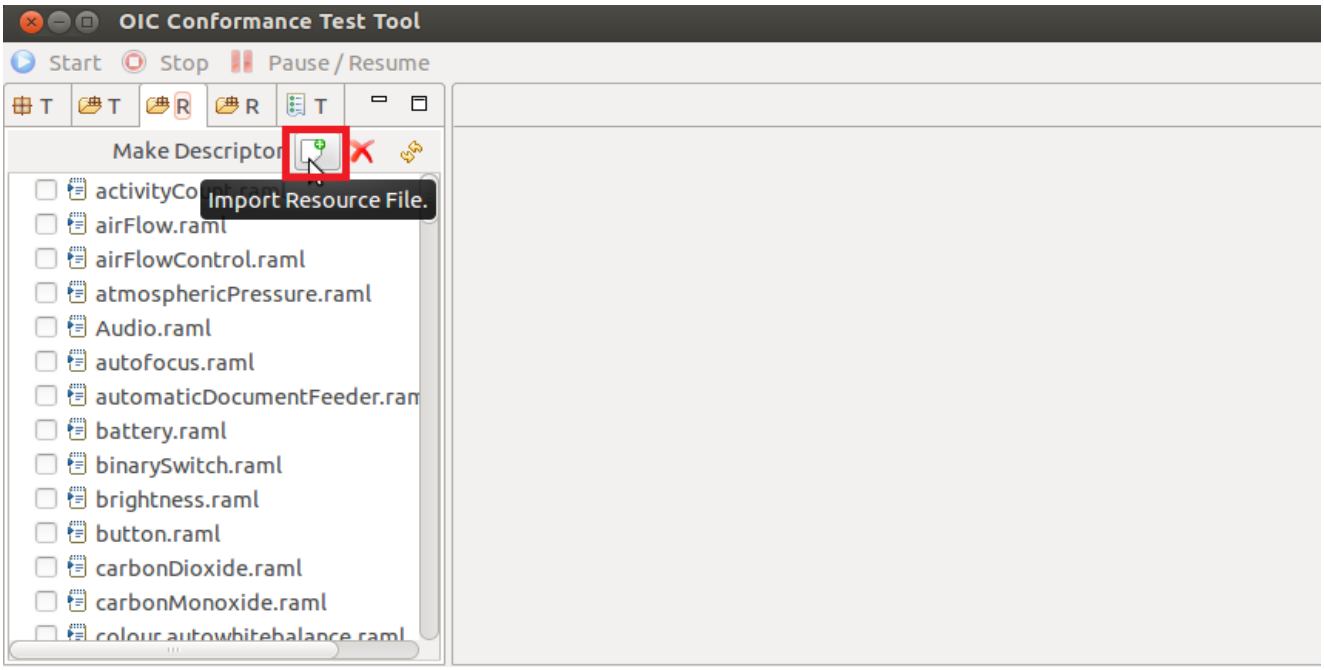

Figure 4.2.7: Conformance Tool Import RAML (Resource) File

- Select RAML file(s) from your local directory and click *OK* button.

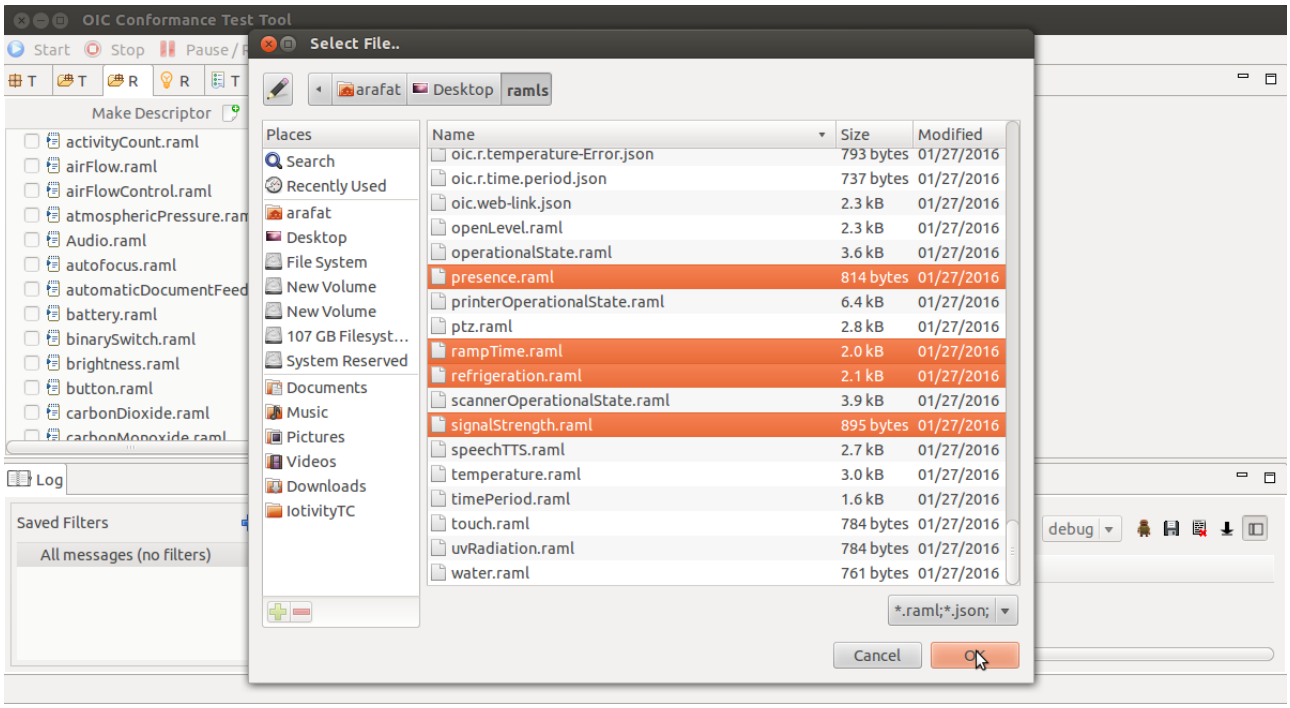

Figure 4.2.8: Select Local RAML File(s)

- To create *DUTDescriptor.json*, select RAML file(s) and Click on *Make Descriptor* button.

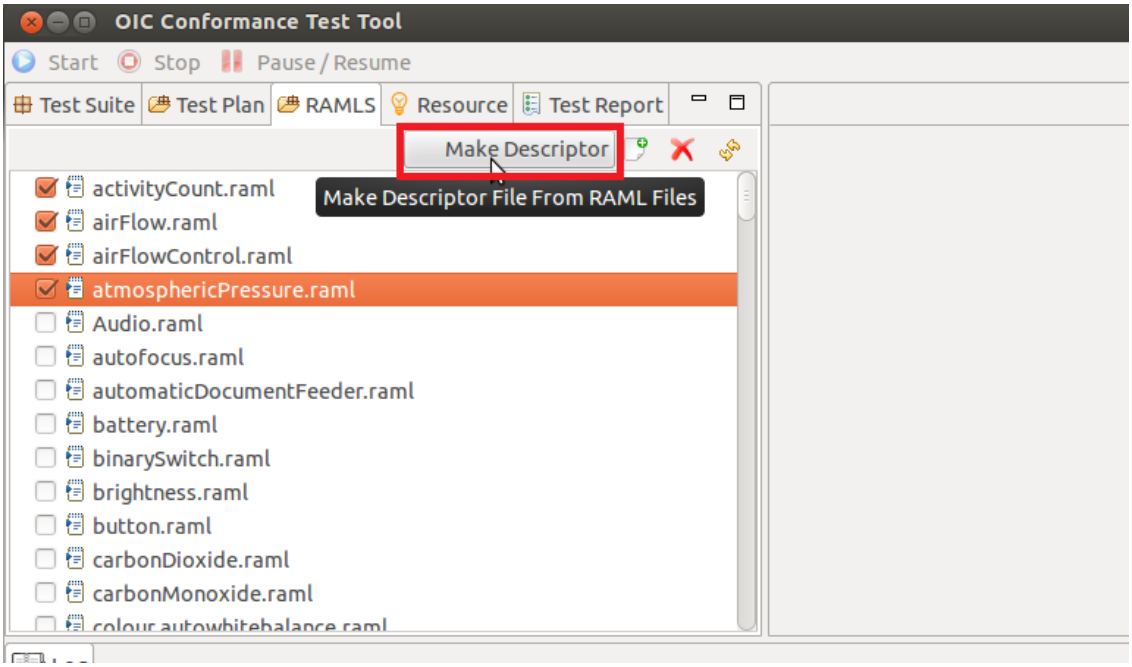

Figure 4.2.9: Make/Create DUTDescriptor.json

- It will create *DUTDescriptor.json* file for selected RAML file(s).
- Go to the Resource View by Clicking on Resource Tab, which shows resources list from *DUTDescriptor.json*.

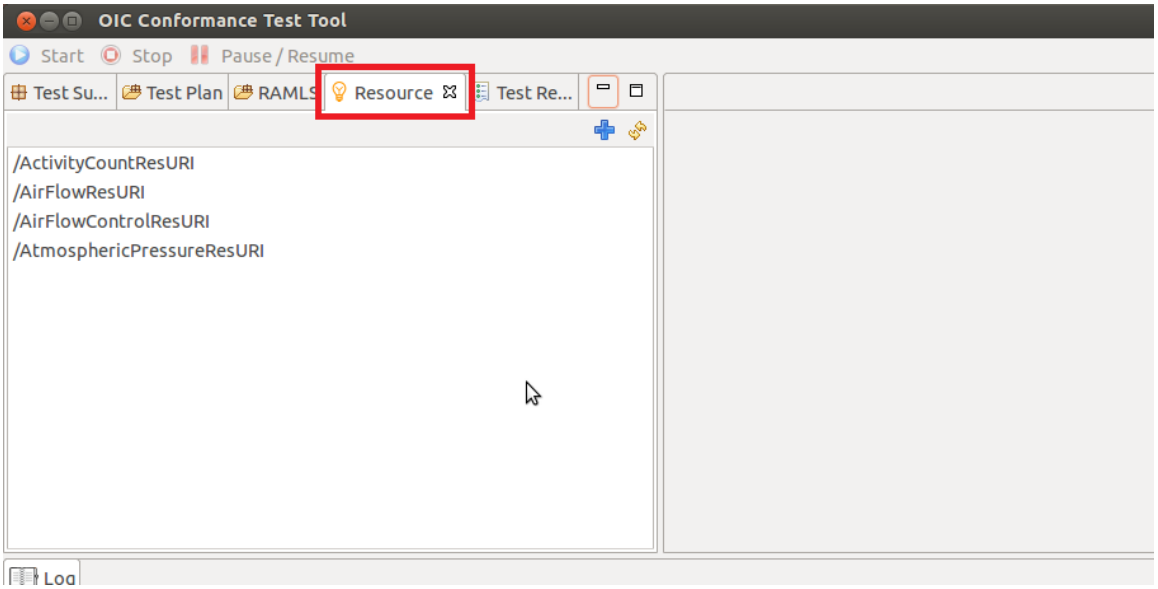

Figure 4.2.10: Resource list of DUTDescriptor

- To create new resource, click on *Create Resource* icon.

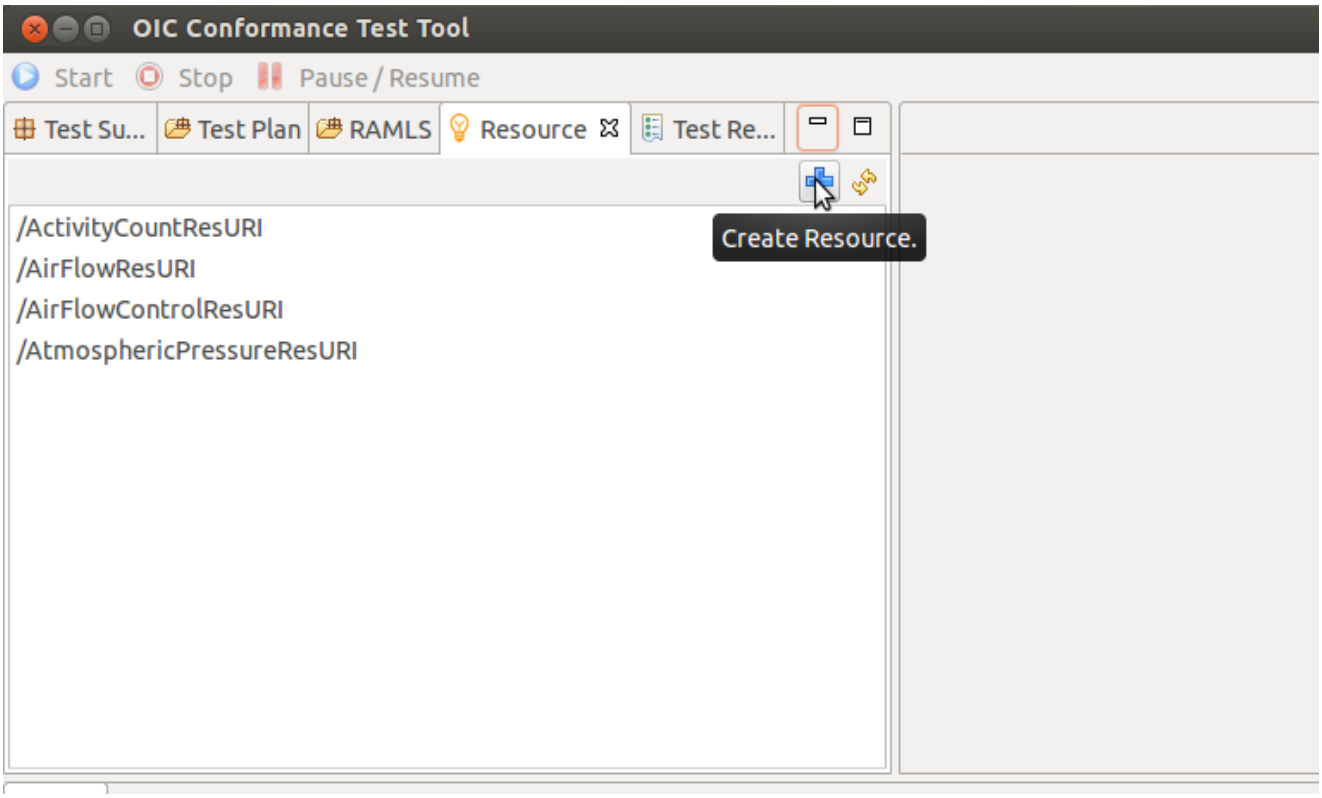

Figure 4.2.11: Create new Resource

- It will open an editor.

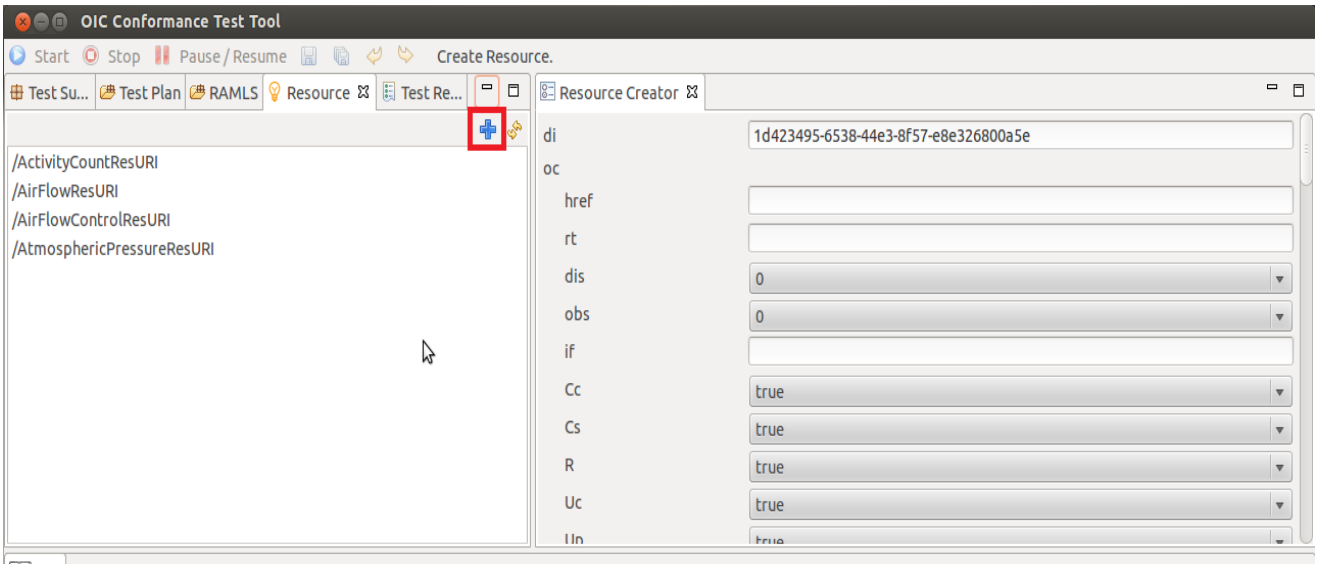

Figure 4.2.12: Resource Editor for creating new resource

- Editor can view as full screen. To do so, double click on *Resource Creator*.

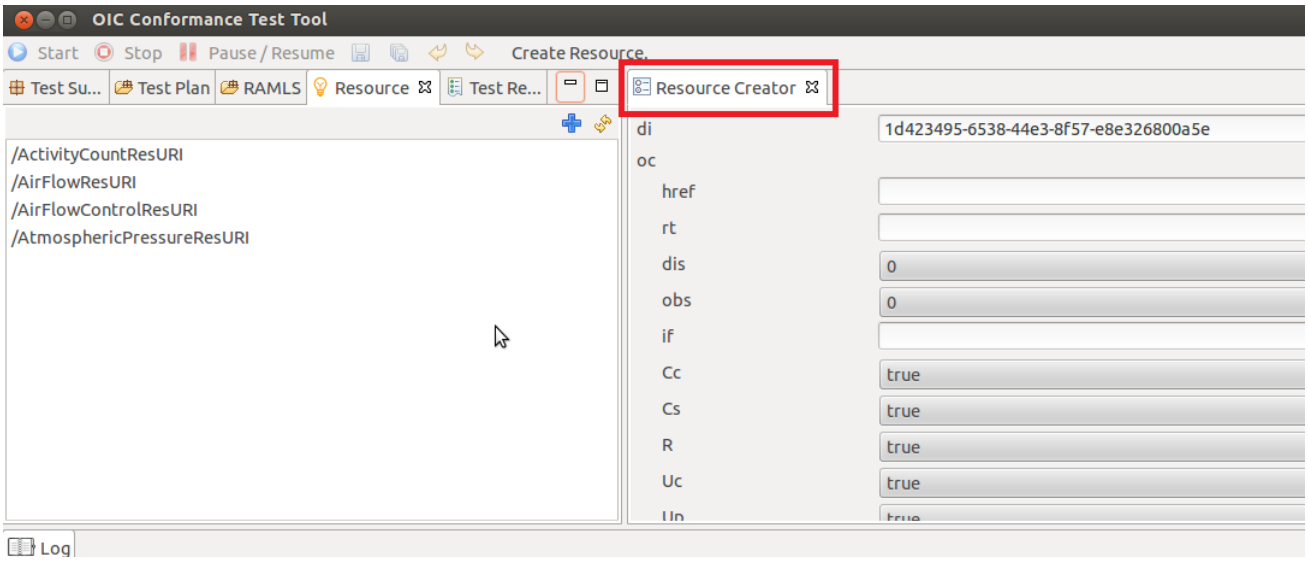

Figure 4.2.13: Opening Creator View in Full Screen

- It will show full screen.

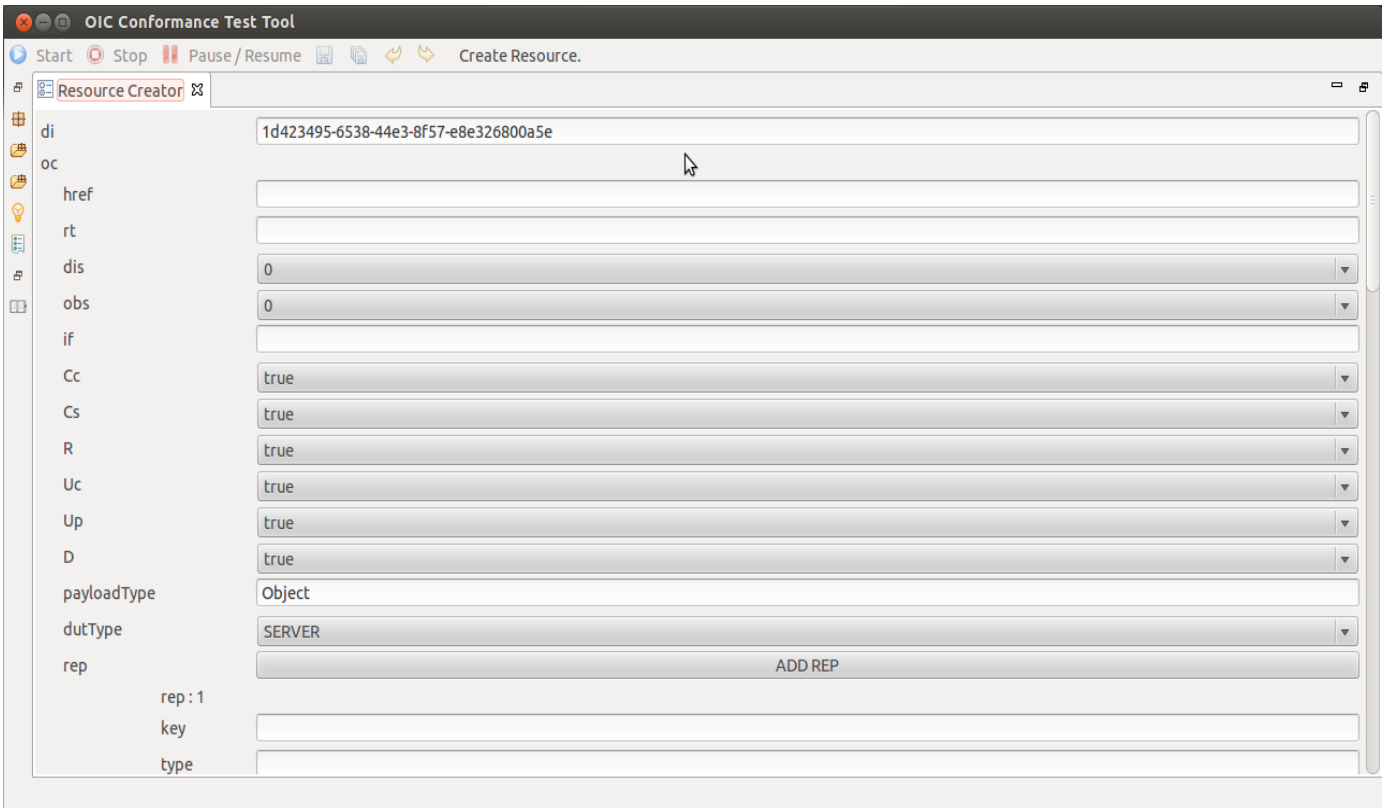

Figure 4.2.14: Full Screen Editor

- Fill all essential fields and click on Create Resource button.

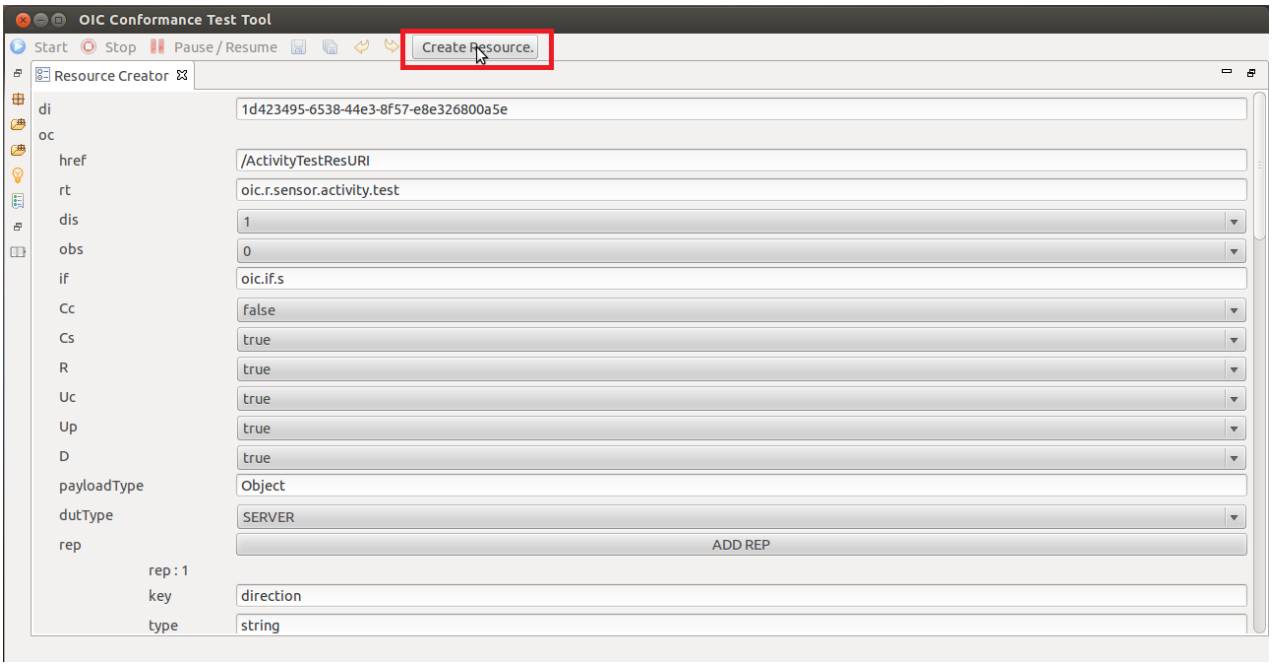

Figure 4.2.15: New Resource Creation

- After finishing, the Resource list will update.

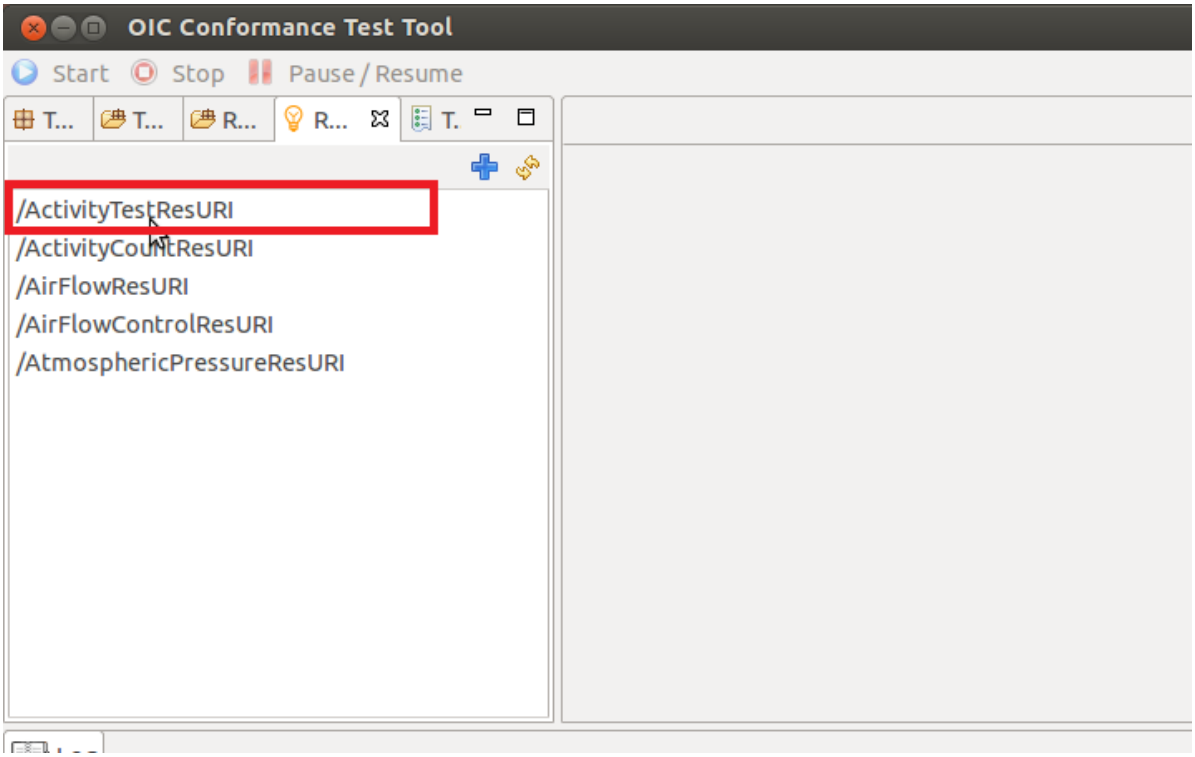

Figure 4.2.16: Resource list after successfully creation of new resource

- You can also add more representation by clicking on *ADD REP* button.

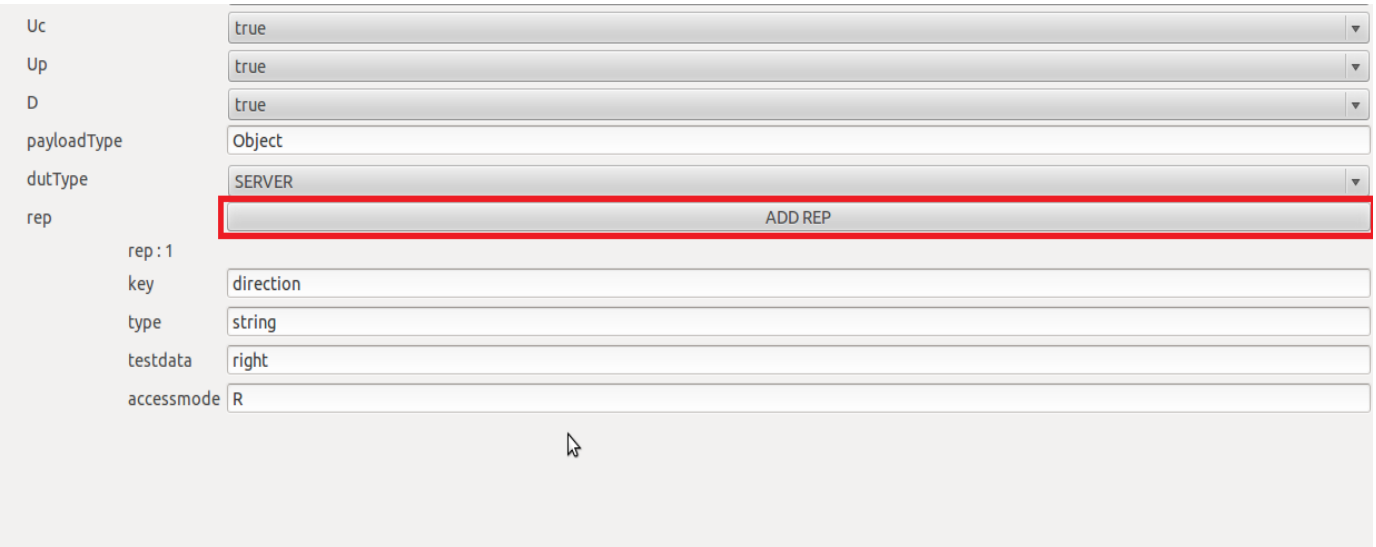

Figure 4.2.17: Add Representation in Resource Description

- New fields will show. Put the representation description and then you can save them.

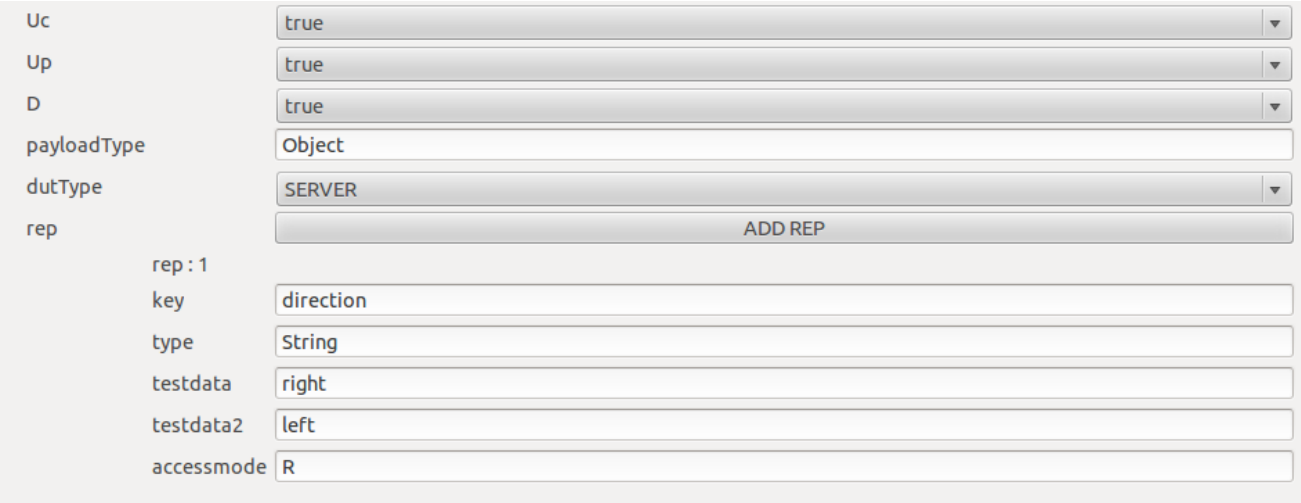

Figure 4.2.18: Representation fields for add new Representation

- To View or Update any resource, double click on that resource from Resource List.

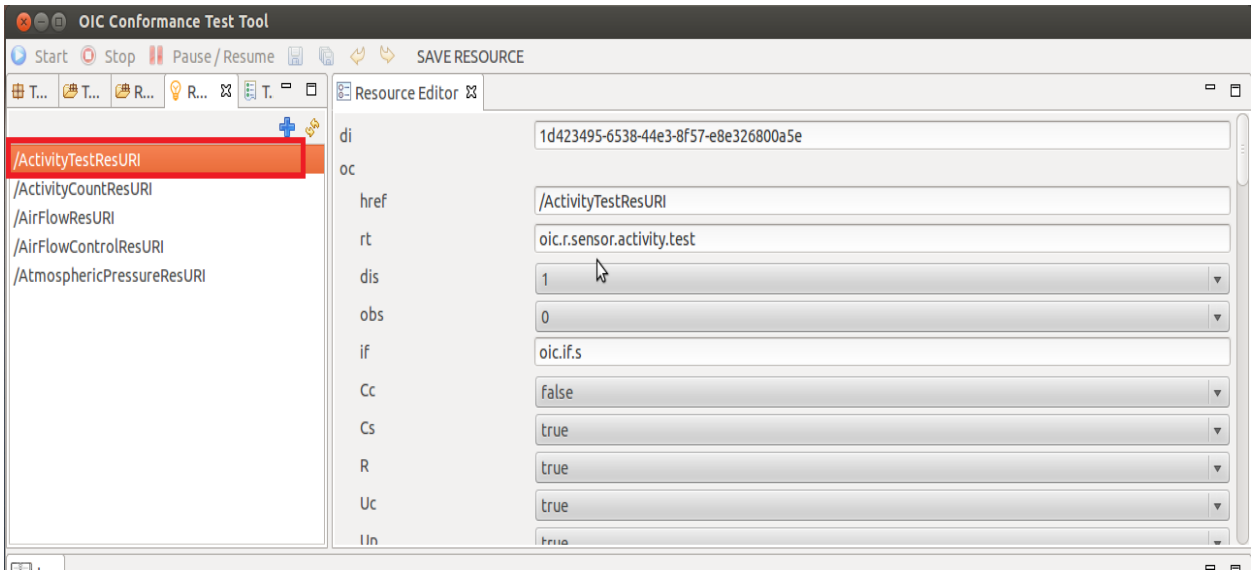

Figure 4.2.19: Show Resource Description

- It will open an editor window with resource description. You can also view the editor in full screen by double clicking on *Resource Editor* which is same as *Resource Creator*.

- Update any description of that resource and save it by clicking on *SAVE RESOURCE* button.

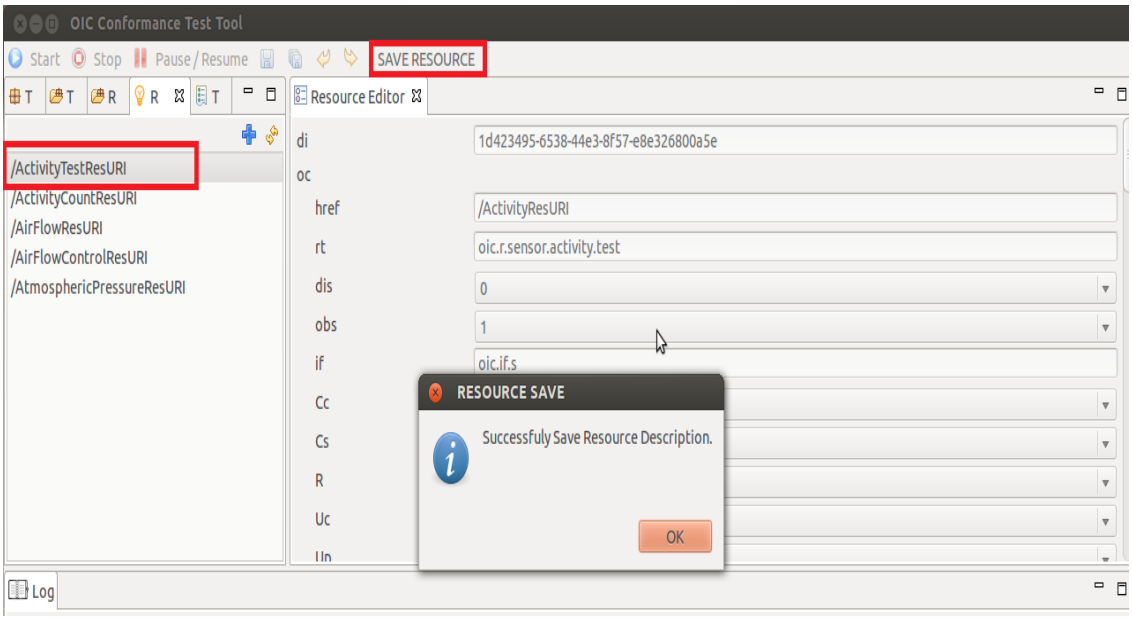

Figure 4.2.20: Updating Resource DUTDescription

- It will save the updated resource description and the list is refreshed with new values and *Resource Editor* view will disappear.

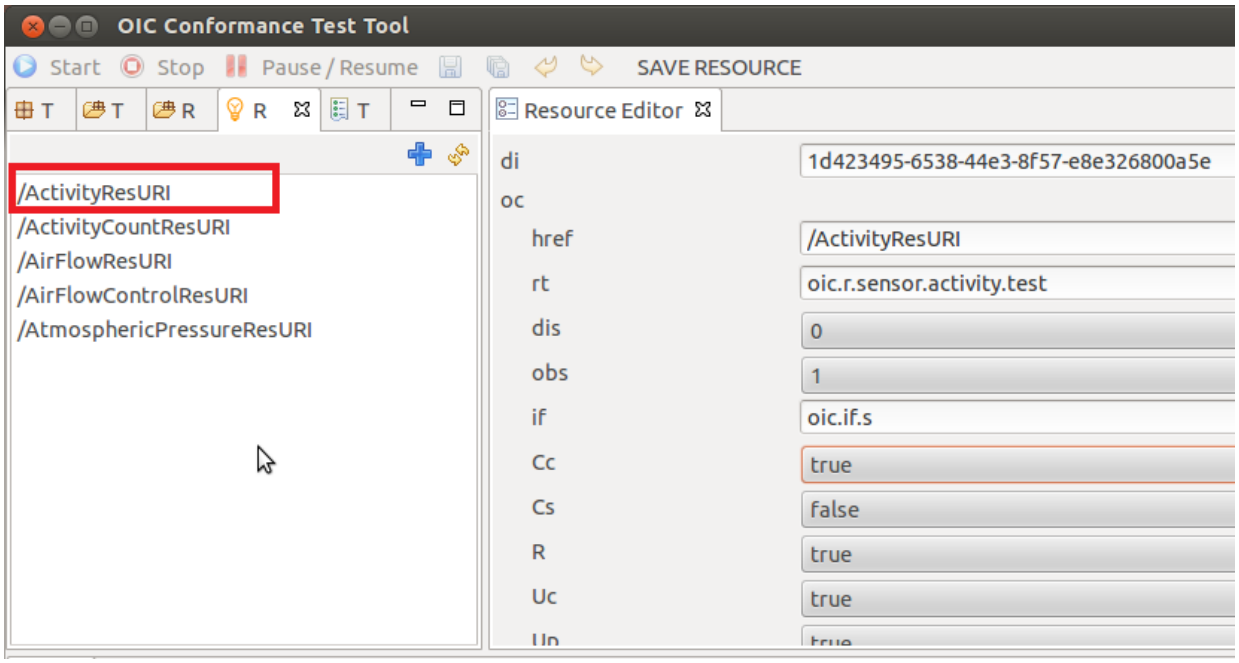

Figure 4.2.21: Saving Updated Resource DUTDescription

- You can also Add Representation, which is described previously (see *Figure 4.2.17* and *Figure 4.2.18*).

g) How to Execute the Test plan

- Once the test plan is selected, click the **Start** button to start the test.

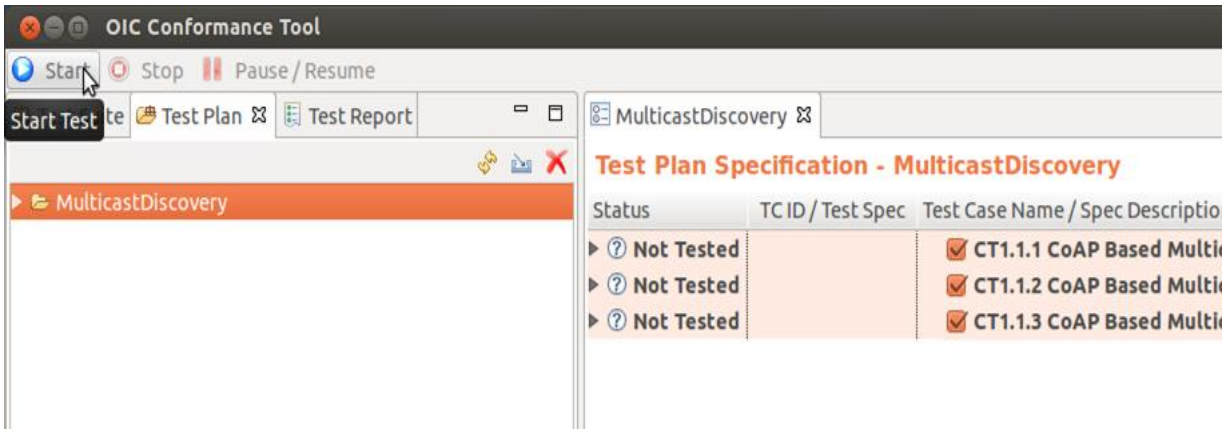

Figure 4.2.22: Conformance Test Tool Start Test Plan

- When the test is running, you can **Stop** or **Pause**/**Resume** the execution. The Flow Graph, Progress & Log windows will show real-time test execution information

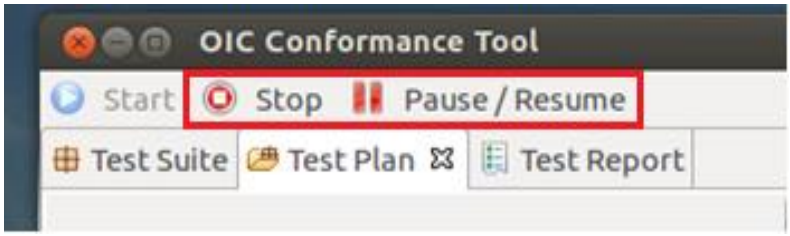

Figure 4.2.23: Stop/Pause Test Plan

h) How to import Test Plan

- Select the Import Test Plan

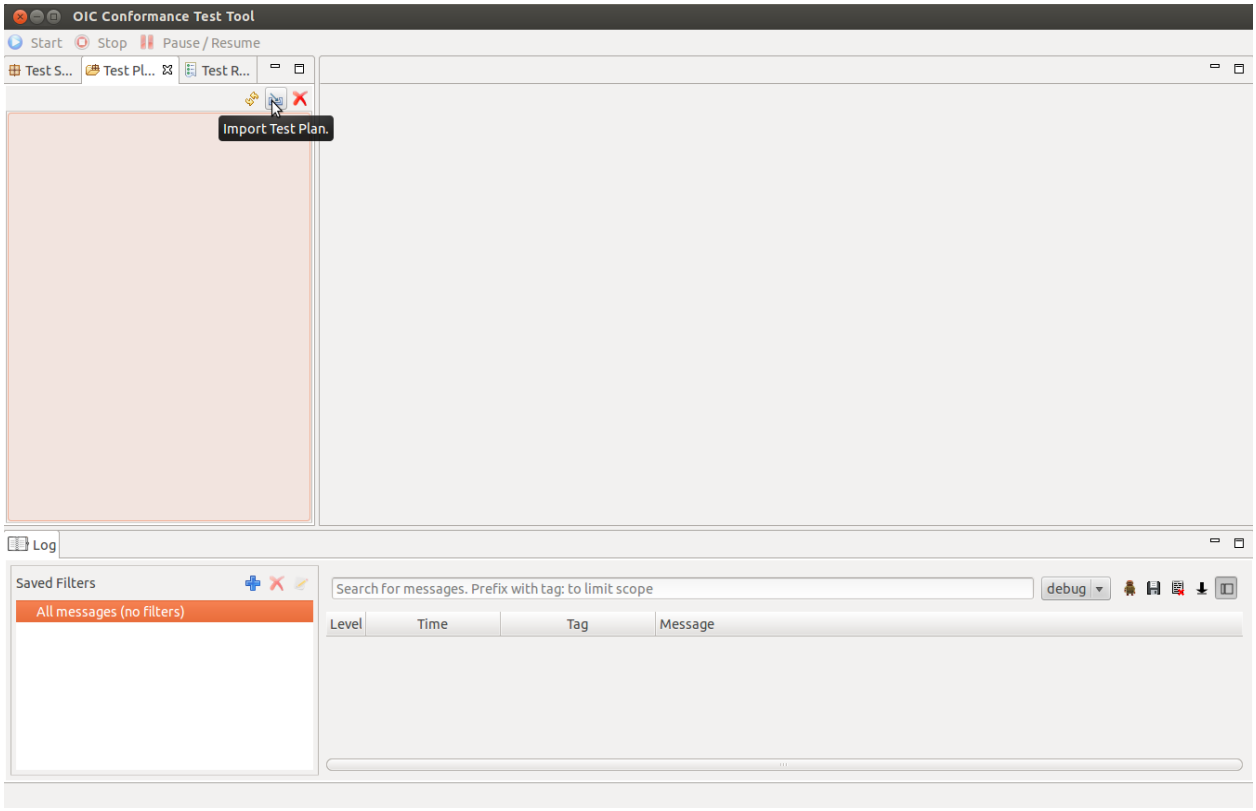

Figure 4.2.24: Import Test Plan

- File browsing window will appear. Go to the directory from where the test plan is to be imported.

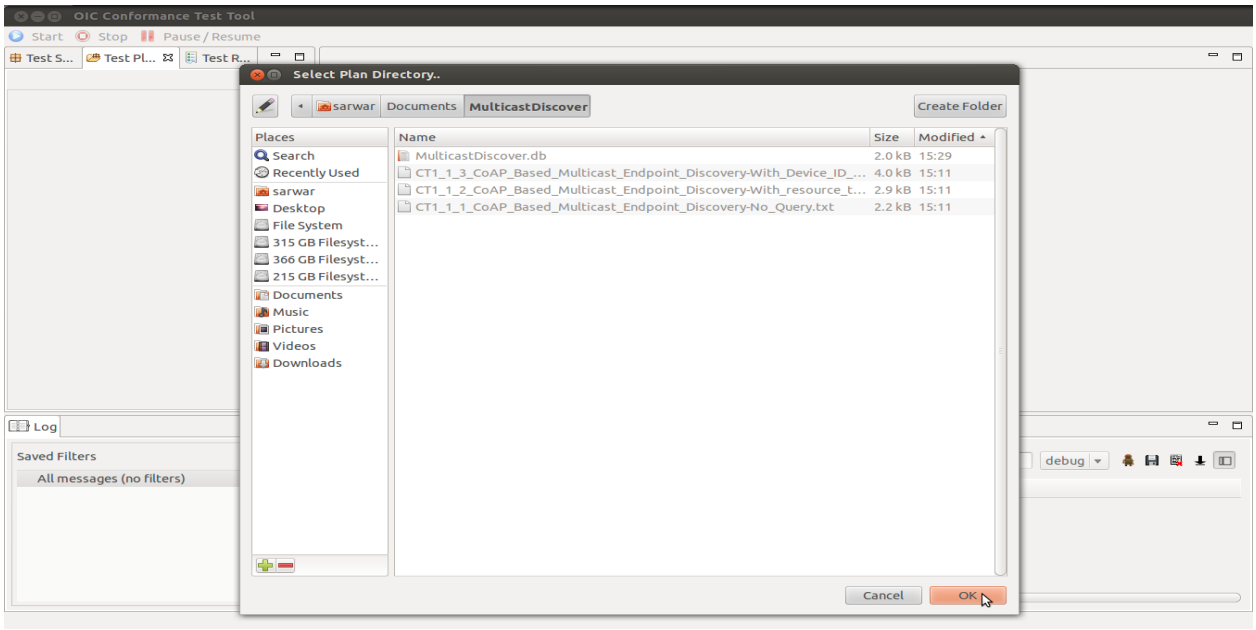

Figure 4.2.25: Select Test Plan Directory

- Then press OK to import.

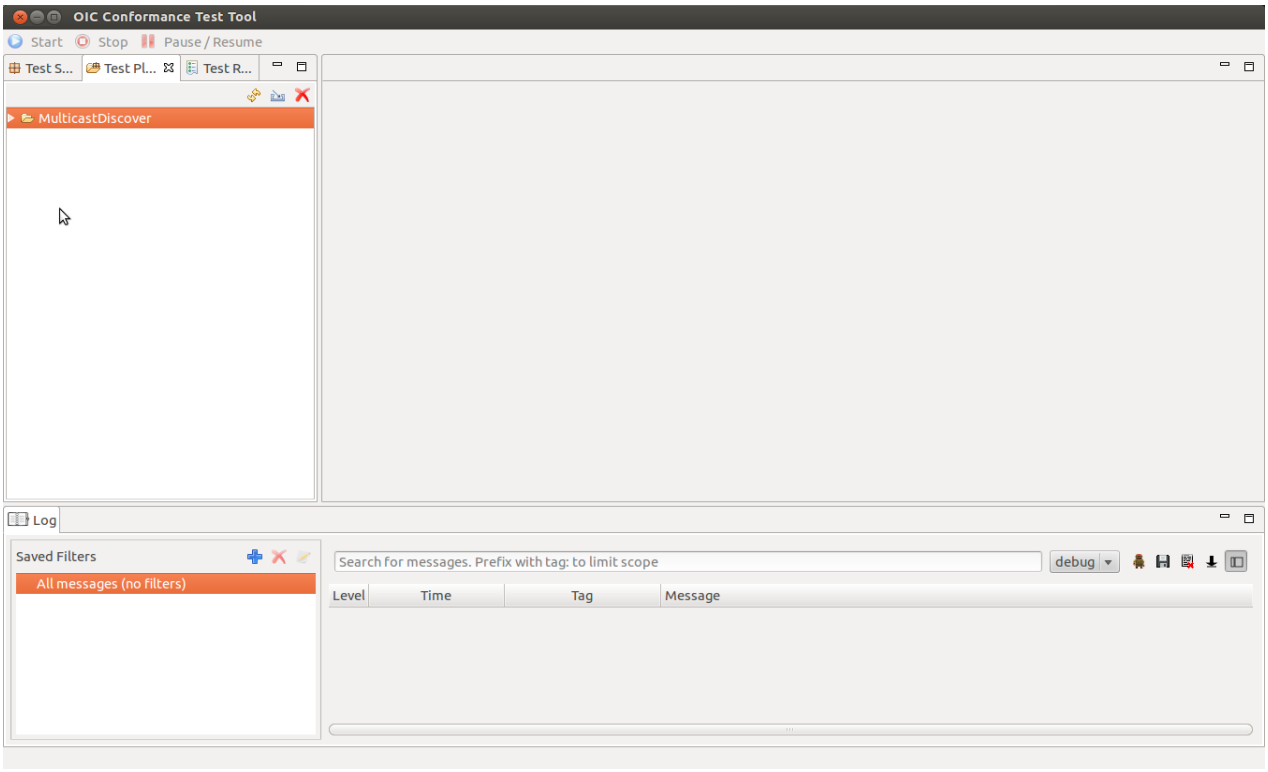

Figure 4.2.26: Imported Test Plan

i) How to Delete Test Suite

➢ Select the Test Suite file(s), which need to be deleted.

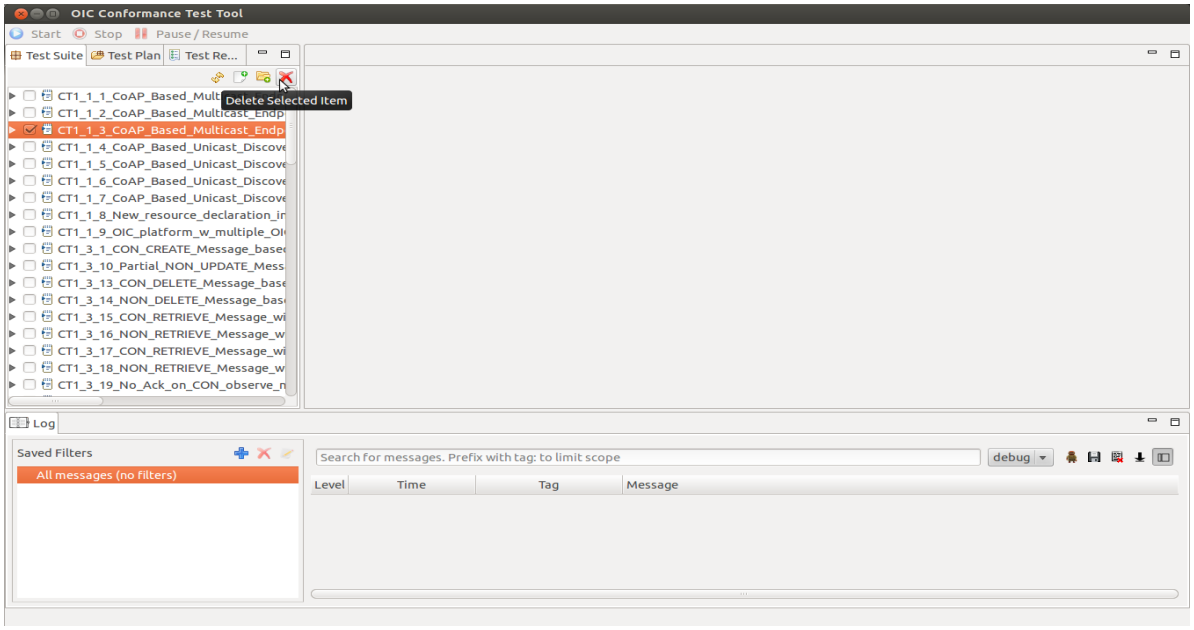

Figure 4.2.27: Select Test Suite(s) for Delete action

➢ Then press delete button. A permission window will popped up.

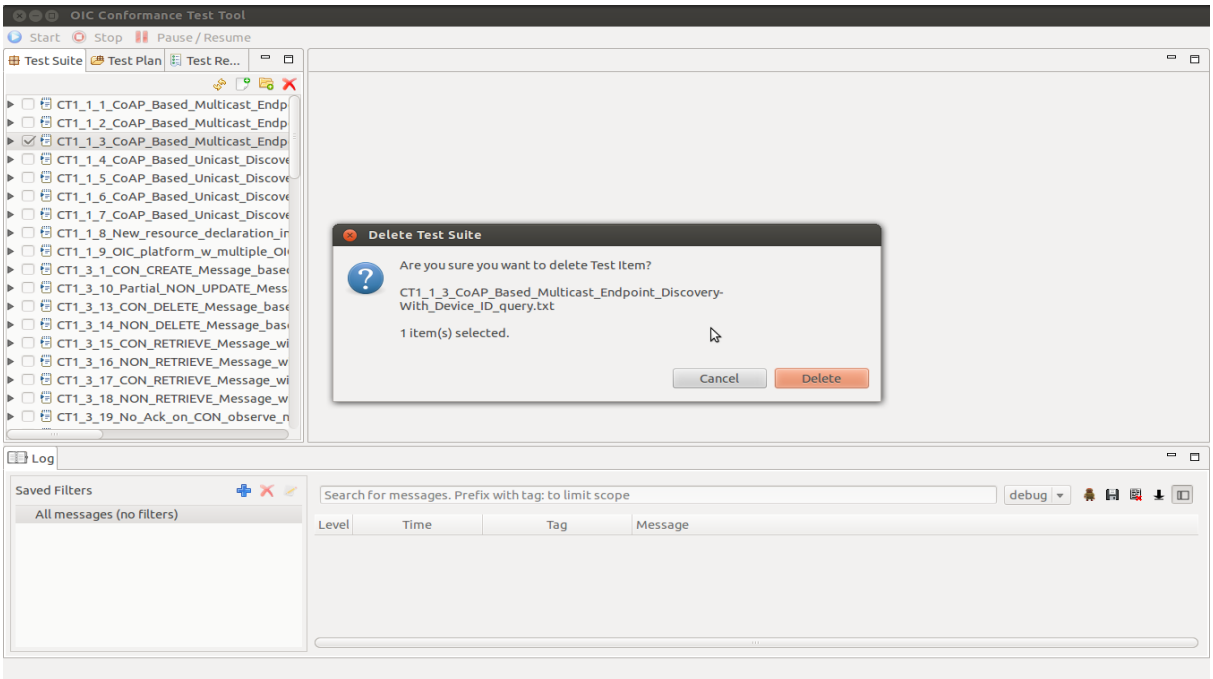

Figure 4.2.28: Delete Test Suite(s)

➢ Select Delete to delete test suite(s).

j) How to view & interpret Reports

- The **Log** tab will show all the detailed logs generated by the test cases.

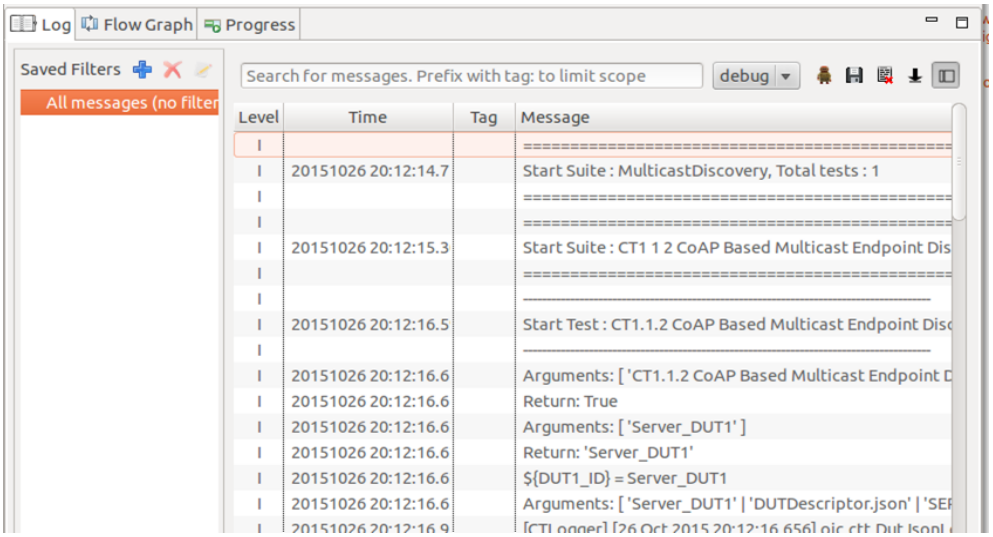

Figure 4.2.29: Conformance Test Tool Test Log

- When the test execution is finished, a **Test Report** window will open with a complete report on the test plan.

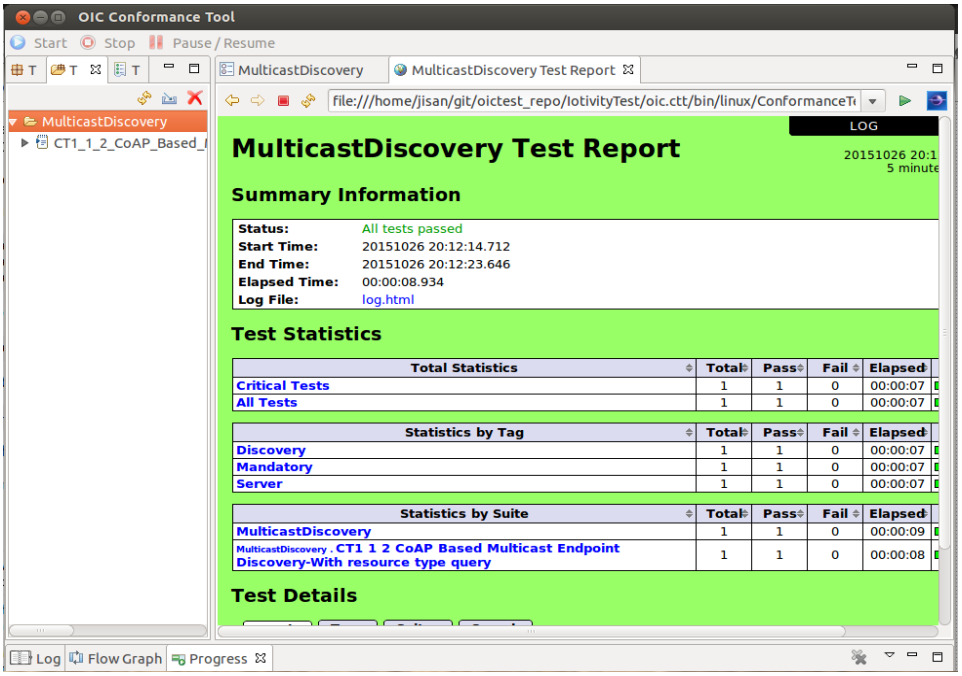

Figure 4.2.30: Conformance Test Tool Test Report

- The reports will be saved in the storage drive. You can view all the generated test reports from the **Test Report** tab.

4.3. Using CLI

a) Go to the root folder of Conformance Tool

b) Then go to the RAMLParser Directory

```
$ cd bin/linux/RAMLParser/bin
```

c) Now run RAMLParser, execute ./RAMLParser with one or more RAML file path as argument, e.g.

```
$ ./RAMLParser ../ramls/airFlow.raml ../ramls/contact.raml
```

d) DUTDescriptor.json will be created and then saved at same directory as ./RAMLParser

e) Copy & Paste this DUTDescriptor.json in *<iotivity_root>/tools/conformance-test-tool*//bin/linux/ConformanceTestTool/libs location

f) Now go back to the root folder of Conformance Tool

g) Then go to the Conformance Test Directory

```
$ cd bin/linux/ConformanceTestTool/ConformanceTool/
testsuite
```

h) To run a TC file, execute *./run_tc.sh testsuite/<TC_File_Name>*, e.g.

```
$ ./run_tc.sh
testsuite/CT1_1_1_CoAP_Based_Multicast_Endpoint_Discovery
-No_Query.txt
```

i) The result report will be saved at < *path_to_conformance_tool>/bin/linux/ ConformanceTestTool/testreport/<timestamp>/*

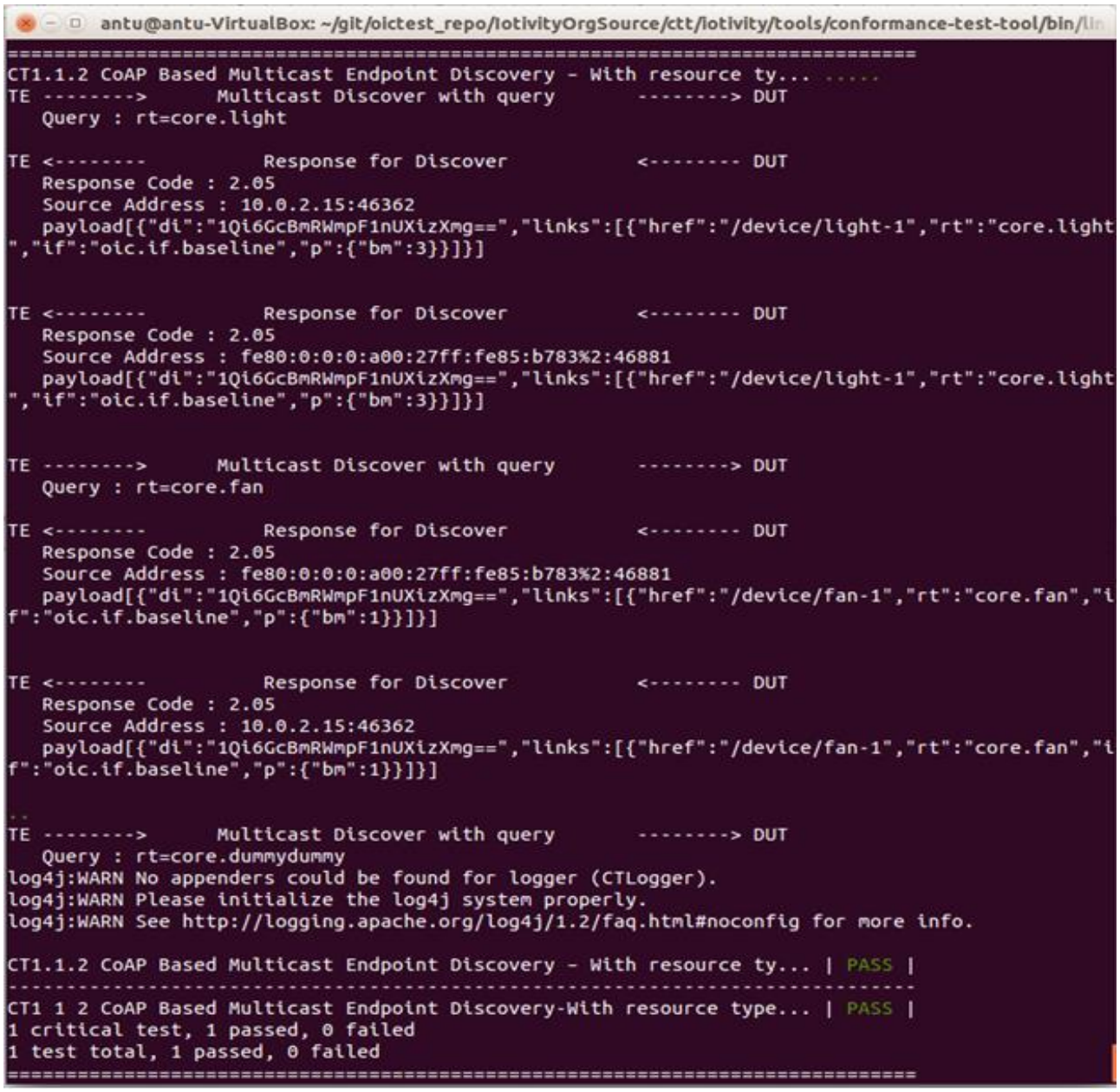

Figure 4.3.1: Execution of a Test Case using Command Line Interface of ConformanceTestTool

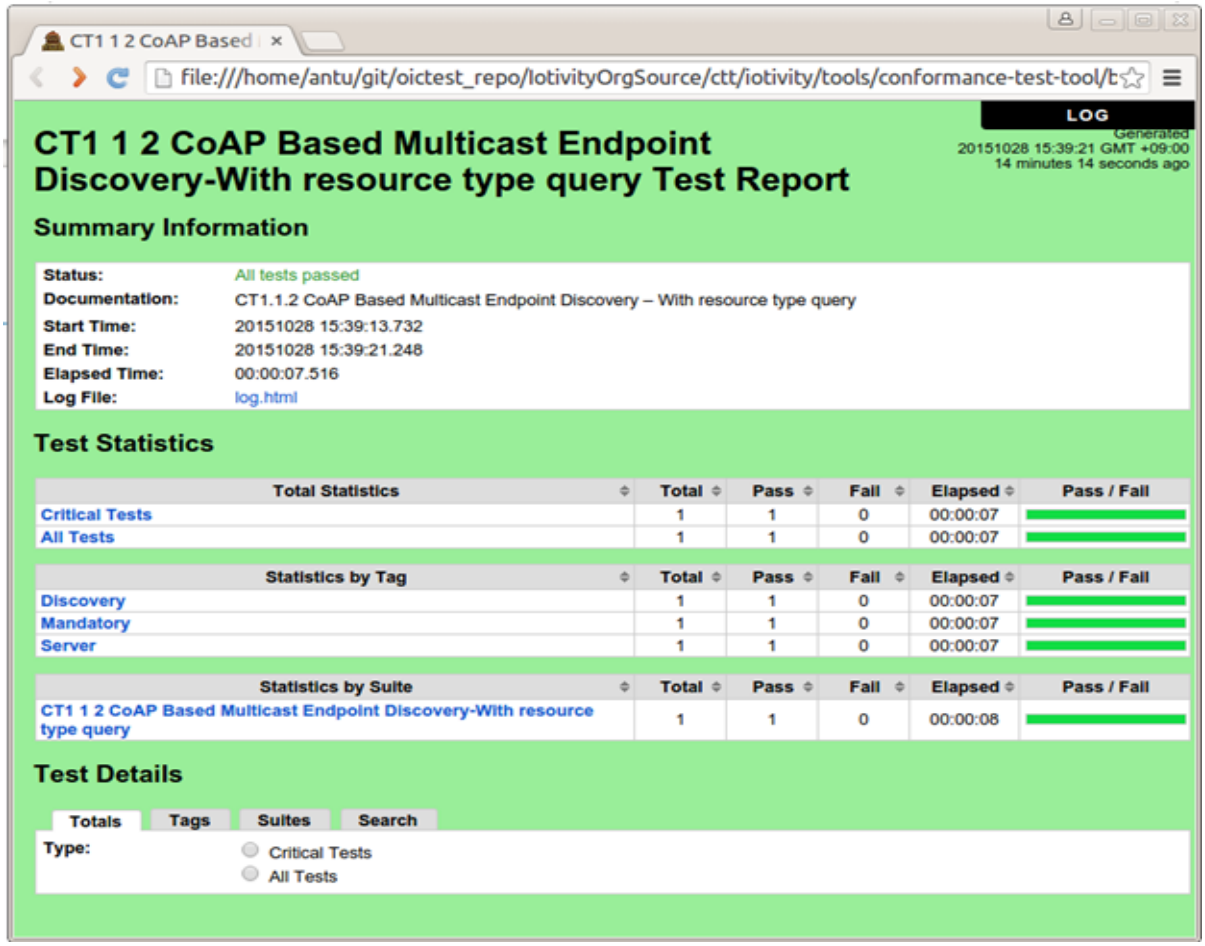

Figure 4.3.2: View of Conformance Test Tool TC Execution Result

4.4. Run Conformance Simulator

- From the top(root) directory of Conformance Test Tool (*<iotivity_root>/tools/conformance-test-tool*), go to linux binary folder:

  `$ cd bin/linux/`

- Append the library location of iotivity to system library path:

  `$ export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:<path-to-iotivity-root>/out/linux/x86/release/`

- To run Conformance Simulator using default settings(Non-conformable, IPv6, non-secure server):

  `$ ./ConformanceSimulator`

- Run Conformance Simulator with customize settings:

```
$ ./ConformanceSimulator [QoS<0/1>] [IP_Version<4/6>]
[Security<0/1/2>]
```

Message Type:
QoS: 0 = NON (Non-Confirmable Message), 1 = CON (Confirmable Message)
IP_Version: 4 = IPv4, 6 = IPv6
Security: 0 = non-secure, 1 = Secure Client, 2 = Secure Server

Example:

```
$ ./ConformanceSimulator 0 4 0
```

It's for, IpV4 non-secure NON-type server

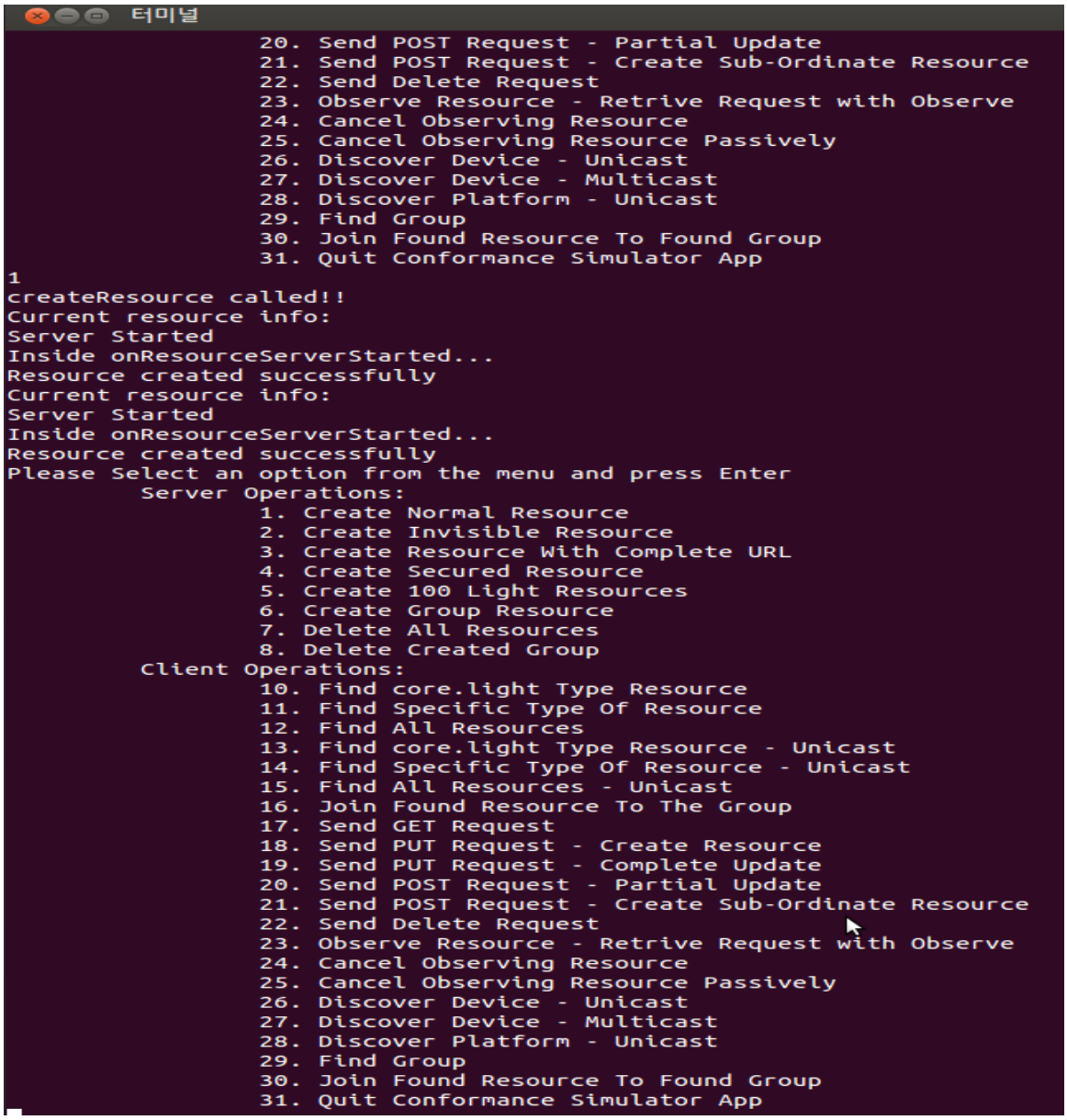

Figure 4.4: Conformance Simulator Execution