Specification of Robot Framework at Bosch

Thomas Pollerspöck

Nguyen Huynh Tri Cuong
Tran Duy Ngoan
Mai Dinh Nam Son
Tran Hoang Nguyen

June 2021

! Information !

Dear reader,

Information to users.

Contents

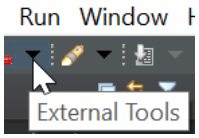
Todo list

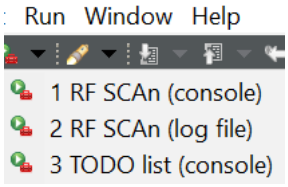
1. Code analysis

The Robotframework AIO installation provides a static code analysis to detect potential errors and violations to coding conventions. The name of the analyser is "Robocop".

Start Eclipse and select a file or a folder in Project Explorer. A missing selection causes an error.

The External Tools menu provides three preconfigured settings to start the analysis (the numbering within the context menu depends on the overall number of External Tools within your Eclipse configuration).





RF SCAn is the abbreviation for "**R**obot **F**ramework **S**tatic **C**ode **An**alysis".

1. **RF SCAn (console)**

   Output printed to console

2. **RF SCAn (log file)**

   Output printed to log file: `%ROBOTLOGPATH%\Robocop.log`

3. **TODO list (console)**

   It is possible to use the string `TODO` in code comments to indicate that still something is *todo*. But before a release all todo's should be resolved (except there is a good and documented reason to keep the marker). The static code analyser can be used to provide a list of all positions within your code, at which still such a `TODO` marker is present.

Hint: The console displays a warning regarding rule '0906'. This warning can be ignored.

2. Library

Document for the library

3. Apertis Pro

3.1 How to use DLT

3.1.1 dlt-daemon on Apertis

Verify that **dlt-daemon** has installed on Apertis Pro target or not.
`systemctl status dlt-daemon`

In case **dlt-daemon** is not available, follow below steps to install and start dlt-daemon service:

- Install **dlt-daemon** package
  `sudo apt install dlt-daemon`
- Start **dlt-daemon** service
  `sudo systemctl start dlt-daemon`

3.1.2 Apertis Pro firewall configuration

In order to capture DLT log/trace from DLT client(**DLT Viewer**, **DLTConnector**), DLT client has to comminucate with Apertis Pro (TCP/IP protocol) via port **3490** (as default).
So that, this connection should be allowed on Apertis Pro target.

Adopt settings of firewall at Apertis Pro:

- Add new rule to allow DLT service at port **3490** (as default)
  Edit `/etc/iptables/rules.v4` file to add below line

```
...
# Accept dlt for development
-A INPUT -p tcp -m state --state NEW -m tcp --dport 3490 -j ACCEPT
...
```

- Restart the firewall with changed parameters
  `sudo systemctl restart iptables.service`

3.1.3 DLTSelfTestApp

DLTSelfTestApp is an application which will be run on the Apertis Pro target for testing the DLT connection between Robotframework AIO and target.
This package is a part of Robotframework Selftest helpers.
To install **DLTSelfTestApp**, download its debian package on Apertis Pro target then execute the below command.
`sudo dpkg -i <path/to/dltselftestapp_1.0.0_amd64.deb>`

DLTSelfTestApp application will be installed in `/opt/bosch/robfw/dlt` directory and can be started with below command:
`/opt/bosch/robfw/dlt/DLTSelfTestApp`

Welcome log message `Welcome to Robotframework AIO DLTSelfTestApp...` will be sent at application startup.
Then the ping log `ping message from Robotframework AIO DLTSelfTestApp` every 5 seconds.

DLT command injection:
To perform the DLT command injection, use below information:

- App ID: **RBFW**
- Context ID: **TEST**
- Service ID: **0x1000**
- Data as Textdata

DLT log reponse of **DLTSelfTestApp** will bases on injected command:

- `welcome` : DLT reponse as above welcome message.
- `exit` : DLT reponse as `Bye...` then **DLTSelfTestApp** will be terminated.
- Other commands: DLT reponse as combination of data and string.
  e.g: `Data: 000000: 77 65 6c 63 6f 6d 65 31 32 31 32 00 xx xx xx xx welcome1212`

3.1.4 QConnectDLTLibrary

QConnectDLTLibrary is part of Robotframework AIO.
It provides the ability for handling connection to Diagnostic Log and Trace(DLT) Module. The library support for getting trace message and sending trace command

Sample Robotframework testcase which are using **QConnectDLTLibrary** to test DLTSelfTestApp on Apertis Pro target:

- Robot `Settings` (Setup, Teardown) and used `Variables` , `Keyword`

```
*** Settings ***
Documentation  This is selftest for DLT connection with DLTSelfTestApp
Library     QConnectionLibrary.ConnectionManager
Suite Setup     Open Connection
Suite Teardown  Close Connection

*** Variables ***
${CONNECTION_NAME}  TEST_CONN_DLTSelfTestApp
${DLT_CONNECTION_CONFIG} =  SEPARATOR=
...  {
...      "gen3flex@DLTLSIMWFH": {
...            "target_ip": "127.0.0.1",
...            "target_port": 4490,
...            "mode": 0,
...            "ecu": "ECU1",
...            "com_port": "COM1",
...            "baudrate": 115200,
...            "server_ip": "localhost",
...            "server_port": 1234
...      }
...  }

*** Keywords ***
Close Connection
   disconnect  ${CONNECTION_NAME}
   Log to console    \nDLT connection has been closed!

Open Connection
   ${dlt_config} =    evaluate    json.loads('''${DLT_CONNECTION_CONFIG}''')    json

   connect             conn_name=${CONNECTION_NAME}
   ...                 conn_type=DLT
   ...                 conn_mode=dltconnector
   ...                 conn_conf=${dlt_config}

   Log to console    \nDLT connection has been opened successfully!
```

- Sample robot testcase to verify the ping message from DLTSelfTestApp

```
*** Test Cases ***
Match log/trace from DLTSelfTestApp
   [Documentation]   Match log/trace from DLTSelfTestApp
   [Tags]   DLTSelfTestApp
   ${res}=    verify     conn_name=${CONNECTION_NAME}
   ...                   search_pattern=(DLT:0x01.*RBFW.*)
   ...                   timeout=6    # DLTSelfTestApp pings a message every 5 seconds

   # log to console     \n${res}[0]
   # verify that reponse message should contain "Ping" keyword
   Should Match Regexp     ${res}[0]    DLT:0x01.*RBFW.*Ping.*
```

- Sample robot testcase to verify command injection with DLTSelfTestApp

```
Command injection with DLTSelfTestApp
   [Documentation]   Get log/trace from DLTSelfTestApp
   [Tags]   DLTSelfTestApp
   ${res}=    verify     conn_name=${CONNECTION_NAME}
   ...                   search_pattern=(DLT:0x01.*RBFW.*Welcome.*)
   ...                   send_cmd=DLT_CALL_SW_INJECTION_ECU ECU1 1000 RBFW TEST 'welcome'

   # log to console     \n${res}[0]

   ${res}=    verify     conn_name=${CONNECTION_NAME}
   ...                   search_pattern=(DLT:0x01.*RBFW.*other_cmd.*)
   ...                   send_cmd=DLT_CALL_SW_INJECTION_ECU ECU1 1000 RBFW TEST 'other_cmd'

   # log to console     \n${res}[0]
```

Please refer QConnectDLTLibrary repository for more details about usage and other example testcase for DLT commection.