Configuring Parasoft For Unit Testing

1. BDF File Generation

- Build the application and deploy it inside the container.
    - *cd home/Vishal*
    - *mkdir WorkFolder*
- Create a WorkFolder and copy the files from container to Workfolder.
    - *docker cp sdk_for_ubuntu:/ap /home/Vishal/Workfolder*
- Do the necessary MakeFile Changes.
- Export the required environment Variables required for running script.
- Add the parasoft build directory in the do_build_linux.sh file.
- Delete or rename the old build file.
- Run script to build the application with Parasoft dependencies
    - *For EVT run do_build_linux.sh with expected params.*
- Give make Command in the build folder, if not added in outer MakeFile.
- Give the executable permission to the application directory.
- Check the BDF file generation inside the build path.

2. Loading the Project in the Parasoft

- Click on import project
- Activate the license passing the IP and test the connection.
- Click on General -> Click on Existing Project.
- Always create a new Workspace and add it to the parasoft while launching the cpptest.sh
- // Give permission chmod 756 to to sample_application/apps/EVTService**/.properties**
- **Project will be loaded.**

3. Generating the Executable File

- Select only src inc and utils for Ut generation. For SAR only src.
- Load the embedded.properties files and start building the executable going in Parasoft -> Embedded System -> generate executable
- Load the unit_test.properties and generate the Unit Tests.
- Give the CHMOD permission in skeleton
- Stop proxy and skeleton containers

- Run other commands to start the skeleton proxy container and deploying the projects.
- Clog and tlog will be generated
- Inside current_tubf in Workspace copy the clog and tlog files. (one time execution will only happen, if we want to run again we will be needed to generate clog and tlog again)
- Load the load_test_report.properties and run it to generate the report.
- Follow the parasoft aubist integration Guide to check more if issues at any place.
- Check in src how many lines are covered.

4. To Create Manual Test Case

- Check in to report how many functionalities are covered or not.
- Click on respective test suite file
- Select Function for which we need to add.
- Give the function name as _Manual_1
- Click on Finish.

- Disable all failed cases, try to add Ut for passed coverage to increase coverage. Failed cases will be represented in red color.
- Later on add one by one the failed case to the coverage and generate the coverage report.

5. When New Code is added.

- Just include the latest changes, delete old build files. And generate the new BDF and follow all previous processes.

6. Pushing the reports to the GIT

- Git init
- Select url {cloning with http}
- Git clone git_path
- Git branch UT_test_report
- Git status
- Git checkout -n branch_name
- Go to respective reports folder
- Add the date for html files that needs to be pushed
- Git commit -m “Add the message”
- Git push origin branch_name
- Git status

7. Fixing SAR Violations

- Load the .properties File for the SAR generation
- Click on Unit test -> test Unititlies -> click on generate SAR report.
- Check whatever SAR reports show the violation, fix it in the code, raise the merge Request.
- Check Again the SAR Reports.
- Don't build and deploy again and again.