QURT Release process

# Introduction

QURT has been released in two different ways.

* VU\_CORE\_KERNEL\_QURT.xx.yy.zz from //source/qcom/qct/core/kernel/Qurt/
* VU\_CORE\_KERNEL\_DEPLOY\_<target>.xx.yy.zz from //deploy/qcom/qct/core/kernel/Qurt/rel/

Starting from QURT 2.3 release, the release process is same for all applications and (or) targets. VU\_CORE\_KERNEL\_QURT.02.03.00 label contains both QURT source files and binary files, source files are located under source/qcom/qct/core/kernel/Qurt/ and binary files are located under //deploy/qcom/qct/core/kernel/Qurt/rel/

# Release process

Some scripts are available under //source/qcom/qct/core/kernel/Qurt/main/latest/test/ to help automate some of the steps. The scripts are verified under linux64 environment. They are not verified under Windows environment.

The following are the steps to make QURT release.

1. Prepare client work space that will reflect the intended Qurt source and test.
2. Prepare work space and environment
   1. export Q6\_RTOS\_SOURCE=<QURT source top>
   2. cd <Qurt test top>
3. Run regression
   1. ./run\_nightly\_regression.bash
   2. Regression takes less than one hour
4. Verify all tests passed
   1. ./verify\_test\_results.bash –s $PWD/build\_regression/test\_logs
   2. It will print PASS/FAIL for each target under the log directory.
   3. ./verify\_test\_results.bash –s $PWD/build\_regression/bitmaskonly\_logs
   4. It will print PASS/FAIL for each target under the log directory.
   5. Make sure there are no failures
5. Prepare the deliverables for //deploy
   1. ./copy\_release.bash –s $PWD/build\_regression/install –d <full path of the directory>
   2. The destination directory should exist
   3. If the deliverables exist already, the script will fail
6. Check in the files prepared above to //deploy
7. Prepare a VU label with source and deploy
8. Prepare test label for the “test” directory. E.g. test label for Rel. 2.3, looks like QURT.02.03.00\_TEST
9. Run tests from the label.
   1. ./run\_release\_test.bash –r <release string in QURT library> -v <VU label> -d <directory to checkout the perforce label>
   2. The script will run tests on each target in VU.
   3. It will check if the version is properly updated in the libraries.
   4. It will print out the test result summary.
10. Run on target testing for critical releases. Prepare an eng. build fill the “SU\_Test\_Request\_Form.docx” form and send it to corebsp.kernel.bvt.
11. Release VU using UM and Package Warehouse
12. Copy “Qurt” created in step 9) to /prj/dsp/qdsp6/releases/rtos/<vu label>. The source directory specified is same the directory specified with “-d” option in step 9)

# Handling patch releases for specific targets

Patch releases to specific targets, some times require branching out in Perforce. These patch releases don’t require releases for other targets under //deploy. The above process adapts to target option “-t” starting from step 5. For example, 9x00 is the intended target for patch release.

1. Prepare the deliverables for //deploy
   1. ./copy\_release.bash –s $PWD/build\_regression/install –d <full path of the directory> -t 9x00
   2. The destination directory should exist
   3. If the deliverables exist already, the script will fail
2. Check in the files prepared above to //deploy
3. Prepare a VU label with source and deploy
4. Prepare test label for the “test” directory. E.g. test label for Rel. 2.3, looks like QURT.02.03.00\_TEST
5. Run tests from the label.
   1. ./run\_release\_test.bash –r <release string in QURT library> -v <VU label> -d <directory to checkout the perforce label> -t 9x00
   2. The script will run tests on each target in VU.
   3. It will check if the version is properly updated in the libraries.
   4. It will print out the test result summary.
6. Run on target testing for critical releases. Prepare an eng. build fill the “SU\_Test\_Request\_Form.docx” form and send it to corebsp.kernel.bvt.
7. Release VU using UM and Package Warehouse
8. Copy “Qurt” created in step 9) to /prj/dsp/qdsp6/release/rtos/<vu label>. The source directory specified is same the directory specified with “-d” option in step 9)