logo

QE Framework – Release Procedure

Andrew Rhyder  
Andrew Starritt

12 February 2018

Copyright (C) 2013, 2014, 2015,2018 Australian Synchrotron.

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.3 or any later version published by the Free Software Foundation;with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts.  
A copy of the license is included in the associated document ‘QE Framework – QEGui and User Interface Design’.

Contents

[Introduction 2](#__RefHeading___Toc184_1630634933)

[Procedure 2](#__RefHeading___Toc186_1630634933)

# Introduction

This document contains details on creating releases of the QE framework. It is intended to be used by those developing and maintaining the QE framework, not GUI or application developers using the QE framework. This procedure specifies the actions to be taken to release the QE framework to the wider EPICS community in additional to those required for a regular internal component release.

This document, as of 12thFebruary 2018, has been significantly updated to reflect the relocation of the public home of QE framework from SourceForge to Github. This document has been internally relocated from the *qeframework* component/repository to the *qtepics.github.io* component/repository.

# Procedure

1. Ensure that you are allowed to commit changes to the qtepics GitHub site, if not please consult site administrator.
2. Ensure all documentation is up to date (.PDFs/.HTML generated from .DOCX for example) and checked into Perforce.
3. Ensure any local changes in Perforce required for the release are commited to the appropriate git repository and pushed to the GitHub site.
4. Run *qtepics.github.io/trunk/tools/qe\_git\_test\_build –all*  within at least one environment to ensure the GitHub head files are at least build-able. Note: a successful build does not mitigate the need for through testing of any new features. If build fails, investigate, fix and restart procedure.
5. Using your preferred Perforce client workspace, checkout (mark for edit) the following file:

//ASP/tec/gui/qeframework/trunk/  
 qeframeworkSup/project/common/QEFrameworkVersion.h

Update version numbers to new release version. Update the QE\_VERSION\_MAJOR, QE\_VERSION\_MINOR, QE\_VERSION\_RELEASE, and QE\_VERSION\_STAGE macro defitions. The later should be set to QE\_VERSION\_STAGE\_PRODUCTION.  
  
Commit this change to Perforce.

1. Using your preferred Perforce client workspace, checkout (mark for edit) the following file:

//ASP/tec/gui/qtepics.github.io/trunk/source/release\_notes.docx

Update the release notes to create a new section for the new release. This will provide a summary of the changes since the last release. While in principle this is available from a number of sources, the easiest is the GitHub logs, e.g. if 3.5.2 is the last release:

git log r3.5.2..HEAD

provides a summary of the changes since the last release (tagged r3.5.2 in this example).  
This is summary can then be used to update the release notes file. Note: this will have to be run for each of the repositories, not just qeframework. However experience to-date is that the changes are mainly within the qeframework repository with some changes in the qegui repository.

Save the modified release notes document and also save as HTML to update the html version.

Commit the updated release notes .docx file to perforce. Perforce is now in a read for release state.

1. Create Perforce component labels, e.g., for each component...

cd …/tec/gui/qeframework  
create\_component\_label qeframework/3-5-3-01 trunk  
create\_release\_label qeframework/3-5-3-01

cd …/tec/gui/qegui  
create\_component\_label qegui/3-5-3-01 trunk  
create\_release\_label qegui/3-5-3-01

etc. etc. for the other components.

Note: the additional -01 suffix. The -NN suffix is used for internal only releases which are subsequently incremented to -02, -03, -04 etc.  
Note: we keep the label aligned.

1. Commit the new files (framework version and release notes docx) to the qeframework and qtepics.github.io repositories. Note: step (c) will have ensured the git repositories are up to date with respect to all other files.
2. Tag *each* git repository (except qeBinaries) with the new release tag.  
   This should be consistent with the new major, minor and release versions specified in QEFrameworkVersion.h as per step (e) above, then push the latests files and tags to the GitHub site, e.g.:

git tag r3.5.3  
git push  
git push –-tags

Note: we tag each repository identically even if there has been no change since the last release.

1. Run *qtepics.github.io/trunk/tools/qe\_git\_test\_build -ffmpeg* on all available environments. This not verifies that the previous commit did not break anything, it now allows you to run  
   *qtepics.github.io/trunk/tools/qe\_generate\_prebuilt* . This will generate a tar archive of the just released versions of qeframework and qegui. The file generated will be of the form:

centos7-qe3.5.3-qt5.9-r3.15.5-linux-x86\_64.tar.gz

Verify that the platform, QE version, Qt version, base release version and the EPICS host architecture parts of the filenamr are as expected. Copy the file to the local qeBinaries git . and add to the repository.

git add centos7-qe3.5.3-qt5.9-r3.15.5-linux-x86\_64.tar.gz  
git commit -m “some message”  
git push

Note: *qe\_git\_test\_build* and *qe\_generate\_prebuilt* is Linux centric. Equivalent Windows tools do not exist yet.

1. email tech-talk.