SCons tool “qt4” - Reference

Philippe Vaucher

Dirk Baechle

2010-07-13

Table of Contents

This reference lists all the variables that are used within the “qt4” tool, and the available builders. It is intended for SCons tool developers and core programmers, as a normal user you should read the manual instead.

1. What it does

The “qt4” tool sets construction variables and registers builders for creating applications and libraries that use the Qt4 framework by Trolltech/Nokia.

It supports the following operations:

1.1. Automatic moc file generation from header files

You do not have to specify moc files explicitly, the tool does it for you. However, there are a few preconditions to do so: Your header file must have the same filebase as your implementation file. It must have one of the suffixes `.h`, `.hpp`, `.H`, `.hxx`, `.hh`. You can turn off automatic moc file generation by setting `QT4_AUTOSCAN` to 0. See also the corresponding builder method `Moc4()`.

1.2. Automatic moc file generation from cxx files

As stated in the Qt documentation, include the moc file at the end of the cxx file. Note that you have to include the file, which is generated by the transformation

```
${QT4_MOCCXXPREFIX}<basename>${QT4_MOCCXXSUFFIX}
```

, by default `<basename>.moc`. A warning is generated after building the moc file, if you do not include the correct file. If you are using VariantDir, you may need to specify `duplicate=1`. You can turn off automatic moc file generation by setting `QT4_AUTOSCAN` to 0. See also the corresponding `Moc4` builder method.

1.3. Handling of .ui files

TODO: describe in a little more detail.

See also the corresponding `Uic4` builder method.

1.4. Handling translation files (.ts and .qm)

TODO: describe in a little more detail.

See also the corresponding builder methods Ts4 and Qm4.

1.5. Compiling resource files (.qrc)

TODO: describe in a little more detail.

See also the corresponding Qrc4 builder method.

2. Builders

2.1. Moc4

Builds an output file from a moc input file. Moc input files are either header files or cxx files. This builder is only available after using the tool 'qt4'.

Example:

```
env.Moc4('foo.h') # generates moc_foo.cc
env.Moc4('foo.cpp') # generates foo.moc
```

2.2. XMoc4

Just like the Moc4 builder, it builds an output file from a moc input file. Moc input files are either header files or cxx files. This builder is only available after using the tool 'qt4'. It is defined separately for the include driven Automoc strategy (#1) and can be controlled via the QT4_XMOC* variables.

Example:

```
env.XMoc4('foo.h') # generates moc_foo.cpp
env.XMoc4('foo.cpp') # generates foo.moc
```

2.3. ExplicitMoc4

Just like the `Moc4` builder, it builds an output file from a moc input file. However, it does not use any default prefix or suffix for the filenames. You can, and have to, specify the full source and target names explicitly. This builder is only available after using the tool 'qt4'. It can be your last resort, when you have to moc single files from/to exotic filenames.

Example:

```
env.ExplicitMoc4('moced_foo.cxx','foo.h') # generates moced_foo.cxx
```

2.4. Uic4

Builds a header file from an .ui file, where the former contains the setup code for a GUI class. This builder is only available after using the tool 'qt4'. Using this builder lets you override the standard naming conventions (be careful: prefixes are always prepended to names of built files; if you don't want prefixes, you may set them to ``).

Example:

```
env.Uic4('foo.ui') # -> 'ui_foo.h'
```

2.5. ExplicitUic4

Just like the `Uic4` builder, it builds a header file from a .ui input file. However, it does not use any default prefix or suffix for the filenames. You can, and have to, specify the full source and target names explicitly. This builder is only available after using the tool 'qt4'. It can be your last resort, when you have to convert .ui files to exotic filenames.

Example:

```
env.ExplicitUic4('uiced_foo.hpp','foo.ui') # generates uiced_foo.hpp
```

2.6. Qrc4

Builds a cxx file, containing all resources from the given `.qrc` file. This builder is only available after using the tool 'qt4'.

Example:

```
env.Qrc4('foo.qrc') # -> ['qrc_foo.cc']
```

2.7. Ts4

Scans the source files in the given path for tr() marked strings, that should get translated. Writes a `.ts` file for the Qt Linguist. This builder is only available after using the tool 'qt4'.

Example:

```
env.Ts4('foo.ts','.') # -> ['foo.ts']
```

2.8. Qm4

Compiles a given `.ts` file (Qt Linguist) into a binary `.qm` file. This builder is only available after using the tool 'qt4'.

Example:

```
env.Qm4('foo.ts') # -> ['foo.qm']
```

3. Variables

QT4DIR	The Qt4 tool tries to read this from the current Environment and `os.environ`. If it is not set and found, the value of QTDIR (in Environment/`os.environ`)

	is used as a fallback. It is used to initialize all QT4_* construction variables listed below.
QT4_AUTOSCAN	The default is '1', which means that the tool is automatically scanning for mocable files (see also `QT4_AUTOSCAN_STRATEGY`). You can set this variable to '0' to switch it off, and then use the `Moc4` Builder to explicitly specify files to run moc on.
QT4_BINPATH	The path where the Qt4 binaries are installed. The default value is '`QT4DIR`/bin'.
QT4_MOCCPPPATH	The path where the Qt4 header files are installed. The default value is '`QT4DIR`/include'. Note: If you set this variable to None, the tool won't change the `CPPPATH` construction variable.
QT4_DEBUG	Prints lots of debugging information while scanning for moc files.
QT4_MOC	The path to the Qt4 moc executable. Default value is '`QT4_BINPATH`/moc'.
QT4_MOCCXXPREFIX	Default value is ''. Prefix for moc output files, when source is a cxx file and the Automoc strategy #0 (Q_OBJECT driven) is used.
QT4_MOCCXXSUFFIX	Default value is '.moc'. Suffix for moc output files, when source is a cxx file and the Automoc strategy #0 (Q_OBJECT driven) is used.
QT4_MOCFROMCXXFLAGS	Default value is '-i'. These flags are passed to moc, when moccing a C++ file.
QT4_MOCFROMHFLAGS	Default value is ''. These flags are passed to moc, when moccing a header file.
QT4_MOCHPREFIX	Default value is 'moc_'. Prefix for moc output files, when the source file is a header and the Automoc strategy #0 (Q_OBJECT driven) is used.
QT4_MOCHSUFFIX	Default value is '`CXXFILESUFFIX`'. Suffix for moc output files, when the source file is a header and the Automoc strategy #0 (Q_OBJECT driven) is used.
QT4_UIC	Default value is '`QT4_BINPATH`/uic'. The path to the Qt4 uic executable.
QT4_UICCOM	Command to generate the required header and source files from .ui form files. Is compiled from `QT4_UIC` and `QT4_UICFLAGS`.
QT4_UICCOMSTR	The string displayed when generating header and source files from .ui form files. If this is not set, then `QT4_UICCOM` (the command line) is displayed.
QT4_UICFLAGS	Default value is ''. These flags are passed to the Qt4 uic executable, when creating header and source files from a .ui form file.
QT4_UICDECLPREFIX	Default value is 'ui_'. Prefix for uic generated header files.
QT4_UICDECLSUFFIX	Default value is '.h'. Suffix for uic generated header files.
QT4_UISUFFIX	Default value is '.ui'. Suffix of designer input files (form files) in Qt4.
QT4_AUTOSCAN_STRATEGY	Default value is '0'. When using the Automoc feature of the Qt4 tool, you can select different strategies for detecting which files should get moced. The simple approach ('0' as the default) scans header and source files for the Q_OBJECT macro, so the trigger 'moc or not' is Q_OBJECT driven. If it is found, the corresponding file gets moced with the `Moc4` builder. This results in the files

'moc_foo.cc' and 'foo.moc' for header and source file, respectively. They get added to the list of sources, for compiling the current library or program. In older Qt manuals, a different technique for mocing is recommended. A cxx file includes the moced output of itself or its header at the end. This approach is somewhat deprecated, but the 'qtsolutions' by Qt are still based on it, for example. You also might have to switch older Qt sources to a new version 4.x.y. Then you can set this variable to '1', for 'include driven' mocing. This means that the tool searches for '#include' statements in all cxx files, containing a file pattern 'moc_foo.cpp' and 'foo.moc' for header and source file, respectively. If the file 'foo.h/foo.cpp' then contains a Q_OBJECT macro, it gets moced but is NOT added to the list of sources. This is the important difference between the two strategies. If no matching include patterns are found for the current cxx file, the Q_OBJECT driven method (#0) is tried as fallback.

QT4_AUTOMOC_SCANCPPPATH
The default is '1', meaning that the tool scans for mocable files not only in the current directory, but also in all CPPPATH folders (see also `QT4_AUTOMOC_CPPPATH`). You can set this variable to '0' to switch it off on rare occasions, e.g. when too many search folders give you a bad performance in large projects.

QT4_AUTOMOC_CPPPATH
The list of paths to scan for mocable files (see also `QT4_AUTOMOC_SCANCPPPATH`), it is empty by default which means that the CPPPATH variable is used. You can set this variable to a subset of CPPPATH in order to improve performance, i.e. to minimize the search space.

QT4_GOBBLECOMMENTS
Default value is '0' (disabled). When you set this variable to '1', you enable the automatic removal of C/C++ comments, while searching for the Q_OBJECT keyword during the Automoc process. This can be helpful if you have the string Q_OBJECT in one of your comments, but don't want this file to get moced.

QT4_CPPDEFINES_PASSTOMOC
Default value is '1' (enabled). When you set this variable to '1', all currently set CPPDEFINES get passed to the moc executable. It does not matter which strategy you selected with `QT4_AUTOSCAN_STRATEGY` or whether you call the `Moc4` builder directly.

QT4_CLEAN_TS
Default value is '0' (disabled). When you set this variable to '1', the `Ts4` builder will delete your .ts files on a 'scons -c'. Normally, these files for the QtLinguist are treated as 'precious' (they are not removed prior to a rebuild) and do not get cleaned.

QT4_XMOCHPREFIX
Default value is 'moc_'. Like `QT4_MOCHPREFIX`, this is the prefix for moc output files, when the source file is a header and the Automoc strategy #1 (include driven) is used.

QT4_XMOCHSUFFIX
Default value is '.cpp'. Like `QT4_MOCHSUFFIX`, this is the suffix for moc output files, when the source file is a header and the Automoc strategy #1 (include driven) is used.

QT4_XMOCCXXPREFIX
Default value is ''. Like `QT4_MOCCXXPREFIX`, this is the prefix for moc output files, when source is a cxx file and the Automoc strategy #1 (include driven) is used.

QT4_XMOCCXXSUFFIX
Default value is '.moc'. Like `QT4_MOCCXXSUFFIX`, this is the suffix for moc output files, when source is a cxx file and the Automoc strategy #1 (include driven) is used.

QT4_RCC	Default value is '`QT4_BINPATH`/rcc'. The path to the Qt4 rcc executable (resource file compiler).
QT4_LUPDATE	Default value is '`QT4_BINPATH`/lupdate'. The path to the Qt4 lupdate executable (updates the .ts files from sources).
QT4_LRELEASE	Default value is '`QT4_BINPATH`/lrelease'. The path to the Qt4 lrelease executable (converts .ts files to binary .qm files).
QT4_QRCFLAGS	Default value is ''. These flags are passed to the Qt4 rcc executable, when compiling a resource file.
QT4_LUPDATEFLAGS	Default value is ''. These flags are passed to the Qt4 lupdate executable, when updating .ts files from sources.
QT4_LRELEASEFLAGS	Default value is ''. These flags are passed to the Qt4 lrelease executable, when compiling .ts files into binary .qm files.
QT4_MOCINCPREFIX	Default value is '-I'. The prefix for specifying include directories to the Qt4 moc executable.
QT4_QRCSUFFIX	Default value is '.qrc'. Suffix of Qt4 resource files.
QT4_QRCCXXSUFFIX	Default value is '$CXXFILESUFFIX'. This is the suffix for compiled .qrc resource files.
QT4_QRCCXXPREFIX	Default value is 'qrc_'. This is the prefix for compiled .qrc resource files.
QT4_MOCINCFLAGS	List of include paths for the Qt4 moc executable, is compiled from `QT4_MOCINCPREFIX`, `QT4_MOCCPPPATH` and `INCSUFFIX`.
QT4_MOCDEFINES	List of CPP defines that are passed to the Qt4 moc executable, is compiled from `QT4_MOCDEFPREFIX`, `CPPDEFINES` and `QT4_MOCDEFSUFFIX`.
QT4_LUPDATECOM	Command to update .ts files for translation from the sources.
QT4_LUPDATECOMSTR	The string displayed when updating .ts files from the sources. If this is not set, then `QT4_LUPDATECOM` (the command line) is displayed.
QT4_LRELEASECOM	Command to convert .ts files to binary .qm files.
QT4_LRELEASECOMSTR	The string displayed when converting .ts files to binary .qm files. If this is not set, then `QT4_RCC` (the command line) is displayed.