## NAME

cmake-qt - CMake Qt Features Reference

## INTRODUCTION

CMake can find and use Qt 4 and Qt 5 libraries. The Qt 4 libraries are found by the **FindQt4** find-module shipped with CMake, whereas the Qt 5 libraries are found using Config-file Packages shipped with Qt 5. See **cmake-packages(7)** for more information about CMake packages, and see *the Qt cmake manual* for your Qt version.

Qt 4 and Qt 5 may be used together in the same **CMake buildsystem**:

```
cmake_minimum_required(VERSION 3.0.0 FATAL_ERROR)

project(Qt4And5)

set(CMAKE_AUTOMOC ON)
set(CMAKE_INCLUDE_CURRENT_DIR ON)

find_package(Qt5Widgets REQUIRED)
add_executable(publisher publisher.cpp)
target_link_libraries(publisher Qt5::Widgets Qt5::DBus)

find_package(Qt4 REQUIRED)
add_executable(subscriber subscriber.cpp)
target_link_libraries(subscriber Qt4::QtGui Qt4::QtDBus)
```

A CMake target may not link to both Qt 4 and Qt 5. A diagnostic is issued if this is attempted or results from transitive target dependency evaluation.

## QT BUILD TOOLS

Qt relies on some bundled tools for code generation, such as **moc** for meta-object code generation, **uic** for widget layout and population, and **rcc** for virtual filesystem content generation. These tools may be automatically invoked by **cmake(1)** if the appropriate conditions are met. The automatic tool invocation may be used with both Qt 4 and Qt 5.

The tools are executed as part of a synthesized custom target generated by CMake. Target dependencies may be added to that custom target by adding them to the **AUTOGEN_TARGET_DEPENDS** target property.

### AUTOMOC

The **AUTOMOC** target property controls whether **cmake(1)** inspects the C++ files in the target to determine if they require **moc** to be run, and to create rules to execute **moc** at the appropriate time.

If a **Q_OBJECT** or **Q_GADGET** macro is found in a header file, **moc** will be run on the file. The result will be put into a file named according to **moc_<basename>.cpp**. If the macro is found in a C++ implementation file, the moc output will be put into a file named according to **<basename>.moc**, following the Qt conventions. The **moc file** may be included by the user in the C++ implementation file with a preprocessor **#include**. If it is not so included, it will be added to a separate file which is compiled into the target.

The **moc** command line will consume the **COMPILE_DEFINITIONS** and **INCLUDE_DIRECTORIES** target properties from the target it is being invoked for, and for the appropriate build configuration.

Generated **moc_*.cpp** and ***.moc** files are placed in the build directory so it is convenient to set the **CMAKE_INCLUDE_CURRENT_DIR** variable. The **AUTOMOC** target property may be pre-set for all following targets by setting the **CMAKE_AUTOMOC** variable. The **AUTOMOC_MOC_OPTIONS** target property may be populated to set options to pass to **moc**. The **CMAKE_AUTOMOC_MOC_OPTIONS** variable may be populated to pre-set the options for all following targets.

## AUTOUIC

The **AUTOUIC** target property controls whether cmake(1) inspects the C++ files in the target to determine if they require **uic** to be run, and to create rules to execute **uic** at the appropriate time.

If a preprocessor **#include** directive is found which matches **ui_<basename>.h**, and a **<basename>.ui** file exists, then **uic** will be executed to generate the appropriate file.

Generated **ui_*.h** files are placed in the build directory so it is convenient to set the **CMAKE_INCLUDE_CURRENT_DIR** variable. The **AUTOUIC** target property may be pre-set for all following targets by setting the **CMAKE_AUTOUIC** variable. The **AUTOUIC_OPTIONS** target property may be populated to set options to pass to **uic**. The **CMAKE_AUTOUIC_OPTIONS** variable may be populated to pre-set the options for all following targets. The **AUTOUIC_OPTIONS** source file property may be set on the **<basename>.ui** file to set particular options for the file. This overrides options from the **AUTOUIC_OPTIONS** target property.

A target may populate the **INTERFACE_AUTOUIC_OPTIONS** target property with options that should be used when invoking **uic**. This must be consistent with the **AUTOUIC_OPTIONS** target property content of the depender target. The **CMAKE_DEBUG_TARGET_PROPERTIES** variable may be used to track the origin target of such **INTERFACE_AUTOUIC_OPTIONS**. This means that a library which provides an alternative translation system for Qt may specify options which should be used when running **uic**:

```
add_library(KI18n klocalizedstring.cpp)
target_link_libraries(KI18n Qt5::Core)

# KI18n uses the tr2i18n() function instead of tr(). That function is
# declared in the klocalizedstring.h header.
set(autouic_options
-tr tr2i18n
-include klocalizedstring.h
)

set_property(TARGET KI18n APPEND PROPERTY
INTERFACE_AUTOUIC_OPTIONS ${autouic_options}
)
```

A consuming project linking to the target exported from upstream automatically uses appropriate options when **uic** is run by **AUTOUIC**, as a result of linking with the **IMPORTED** target:

```
set(CMAKE_AUTOUIC ON)
# Uses a libwidget.ui file:
add_library(LibWidget libwidget.cpp)
target_link_libraries(LibWidget
KF5::KI18n
Qt5::Widgets
)
```

## AUTORCC

The **AUTORCC** target property controls whether cmake(1) creates rules to execute **rcc** at the appropriate time on source files which have the suffix **.qrc**.

```
add_executable(myexe main.cpp resource_file.qrc)
```

The **AUTORCC** target property may be pre-set for all following targets by setting the **CMAKE_AUTORCC** variable. The **AUTORCC_OPTIONS** target property may be populated to set options to pass to **rcc**. The **CMAKE_AUTORCC_OPTIONS** variable may be populated to pre-set the options for all following targets. The **AUTORCC_OPTIONS** source file property may be set on the **<name>.qrc** file to set particular options for the file. This overrides options from the **AUTORCC_OPTIONS** target property.

## QTMAIN.LIB ON WINDOWS

The Qt 4 and 5 **IMPORTED** targets for the QtGui libraries specify that the qtmain.lib static library shipped with Qt will be linked by all dependent executables which have the **WIN32_EXECUTABLE** enabled.

To disable this behavior, enable the **Qt5_NO_LINK_QTMAIN** target property for Qt 5 based targets or **QT4_NO_LINK_QTMAIN** target property for Qt 4 based targets.

```
add_executable(myexe WIN32 main.cpp)
target_link_libraries(myexe Qt4::QtGui)

add_executable(myexe_no_qtmain WIN32 main_no_qtmain.cpp)
set_property(TARGET main_no_qtmain PROPERTY QT4_NO_LINK_QTMAIN ON)
target_link_libraries(main_no_qtmain Qt4::QtGui)
```

## COPYRIGHT