

CMake Variable Reference

Module variables

Qt modules loaded with `find_package` set various variables.

Note: You rarely need to access these variables directly. Common tasks like linking against a module should be done through the library targets each module defines.

For example, `find_package(Qt6 COMPONENTS Widgets)`, when successful, makes the following variables available:

Variable	Description
Qt6Widgets_COMPILE_DEFINITIONS	A list of compile definitions to use when building against the library.
Qt6Widgets_DEFINITIONS	A list of definitions to use when building against the library.
Qt6Widgets_EXECUTABLE_COMPILE_FLAGS	A string of flags to use when building executables against the library.
Qt6Widgets_FOUND	A boolean that describes whether the module was found successfully.
Qt6Widgets_INCLUDE_DIRS	A list of include directories to use when building against the library.
Qt6Widgets_LIBRARIES	The name of the imported target for the module: Qt5::Widgets
Qt6Widgets_PRIVATE_INCLUDE_DIRS	A list of private include directories to use when building against the library and using private Qt API.
Qt6Widgets_VERSION_STRING	A string containing the module's version.

For all packages found with `find_package`, equivalents of these variables are available; they are case-sensitive.

Installation variables

Additionally, there are also variables that don't relate to a particular package, but to the Qt installation itself.

	<p>forward to in case of mixed Qt 5 and Qt 6 projects. It needs to be set to either 5 or 6 before the respective <code>find_package()</code> calls.</p> <p>If set to 5, commands starting with <code>qt_</code> will call their counterpart starting with <code>qt5_</code>. If set to 6, they will call their counterpart starting with <code>qt6_</code>.</p> <p>If not set, the first <code>find_package</code> call defines the default version.</p>
<code>QT_LIBINFIX</code>	A string that holds the infix used in library names, when Qt is configured with <code>-libinfix</code> .
<code>QT_NO_CREATE_VERSIONLESS_FUNCTIONS</code>	Hides commands that start with <code>qt_</code> , leaving only the versioned ones starting with <code>qt6_</code> .
<code>QT_NO_CREATE_VERSIONLESS_TARGETS</code>	Hides the imported targets starting with <code>Qt : : </code> . Instead, you need to use the targets starting with <code>Qt6 : : </code> .
<code>QT_VISIBILITY_AVAILABLE</code>	On Unix, a boolean that describes whether Qt libraries and plugins were compiled with <code>-fvisibility=hidden</code> . This means that only selected symbols are exported.

Project variables

These variables can influence CMake commands provided by Qt. They may be set by the project, a toolchain file or other third-party packages.

Qt6::Core

<code>ANDROID_NDK_HOST_SYSTEM_NAME</code>	Android-specific architecture of the host system
<code>ANDROID_SDK_ROOT</code>	Location of the Android SDK
<code>QT_ANDROID_ABIS</code>	List of ABIs that the project packages are built for
<code>QT_ANDROID_APPLICATION_ARGUMENTS</code>	List of arguments to pass to Android applications
<code>QT_ANDROID_BUILD_ALL_ABIS</code>	Enables building multi-ABI packages using the autodetected Qt for Android SDK list
<code>QT_ANDROID_SIGN_AAB</code>	Sign the .aab package with the specified keystore, alias and store password
<code>QT_ANDROID_SIGN_APK</code>	Sign the package with the specified keystore, alias and store password
<code>QT_DEPLOY_BIN_DIR</code>	Prefix-relative subdirectory for deploying runtime binaries on some target platforms
<code>QT_DEPLOY_LIB_DIR</code>	Prefix-relative subdirectory for deploying libraries on some target platforms
<code>QT_DEPLOY_PLUGINS_DIR</code>	Prefix-relative subdirectory for deploying Qt plugins on some target platforms
<code>QT_DEPLOY_PREFIX</code>	Base location for a deployment
<code>QT_DEPLOY_QML_DIR</code>	Prefix-relative subdirectory for deploying QML plugins on some target platforms

QT_HOST_PATH	Location of the host Qt installation when cross-compiling
QT_IOS_LAUNCH_SCREEN	Path to iOS launch screen storyboard used by all targets
QT_NO_COLLECT_BUILD_TREE_APK_DEPS	Prevents collecting of project-built shared library targets during Android deployment
QT_NO_SET_XCODE_BUNDLE_IDENTIFIER	Disables providing a fallback app bundle ID during target finalization on iOS
QT_NO_SET_XCODE_DEVELOPMENT_TEAM_ID	Disables providing a fallback team ID during target finalization on iOS
QT_NO_STANDARD_PROJECT_SETUP	Prevents subsequent calls to qt_standard_project_setup() from making any changes
QT_PATH_ANDROID_ABI_<ABI>	Set of variables to specify the path to Qt for Android for the corresponding ABI

Qt6::Qml

QT_QML_OUTPUT_DIRECTORY	Base output directory below which QML modules will be created by default
-------------------------	--

Qt6::InterfaceFramework

< CMake Command Reference

CMake Property Reference >

© 2022 The Qt Company Ltd. Documentation contributions included herein are the copyrights of their respective owners. The documentation provided herein is licensed under the terms of the [GNU Free Documentation License version 1.3](#) as published by the Free Software Foundation. Qt and respective logos are [trademarks](#) of The Qt Company Ltd. in Finland and/or other countries worldwide. All other trademarks are property of their respective owners.



Contact Us

Company

- About Us
- Investors
- Newsroom
- Careers
- Office Locations

Licensing

- Terms & Conditions
- Open Source
- FAQ



Support Services
Professional Services
Partners
Training

Support Center
Downloads
Qt Login
Contact Us
Customer Success

Community

Contribute to Qt
Forum
Wiki
Downloads
Marketplace

© 2022 The Qt Company

[Feedback](#) [Sign In](#)