

[Qt 6.4](#) > [Build with CMake](#) > [Qt 5 and Qt 6 compatibility](#).

Qt 5 and Qt 6 compatibility

The semantics of the CMake API in Qt 5 and Qt 6 are largely compatible. However, up to Qt 5.14, all imported Qt library targets and commands contained the version number as part of the name. This makes writing CMake code that should work with both Qt 5 and Qt 6 somewhat cumbersome. Qt 5.15 therefore introduced *versionless* targets and commands to enable writing CMake code that is largely agnostic to the different Qt versions.

Versionless targets

In addition to the existing imported targets, Qt 5.15 introduced *versionless* targets. That is, to link against [Qt Core](#) one can both reference `, or :Qt6::CoreQt::Core`

```
find_package(Qt6 COMPONENTS Core)
if (NOT Qt6_FOUND)
    find_package(Qt5 5.15 REQUIRED COMPONENTS Core)
endif()

add_executable(helloworld
    ...
)

target_link_libraries(helloworld PRIVATE Qt::Core)
```

Above snippet first tries to find a Qt 6 installation. If that fails, it tries to find a Qt 5.15 package. Independent of whether Qt 6 or Qt 5 is used, we can use the imported target `Qt::Core`

The versionless targets are defined by default. Set `QT_NO_CREATE_VERSIONLESS_TARGETS` before the first call to disable them. `find_package()`

Note: The imported `Qt::Core` target will not feature the target properties that are available in the `Qt6::Core` target.

Versionless commands

Since Qt 5.15, the Qt modules also provide versionless variants of their [commands](#). You can for instance now use `qt_add_translation` to compile translation files, independent of whether you use Qt 5 or Qt 6.

Mixing Qt 5 and Qt 6

There might be projects that need to load both Qt 5 and Qt 6 in one CMake context (though mixing Qt versions in one library or executable is not supported, so be careful there).

In such a setup the versionless targets and commands will be implicitly referring to the first Qt version that was found via `find_package`. Set the `QT_DEFAULT_MAJOR_VERSION` CMake variable before the first call to make the version explicit.

Supporting older Qt 5 versions

If you need to support also Qt 5 versions older than Qt 5.15, you can do so by storing the current version in an CMake variable:

```
find_package(QT NAMES Qt6 Qt5 REQUIRED COMPONENTS Core)
find_package(Qt${QT_VERSION_MAJOR} REQUIRED COMPONENTS Core)

add_executable(helloworld
    ...
)

target_link_libraries(helloworld PRIVATE Qt${QT_VERSION_MAJOR}::Core)
```

Here we let try to find first Qt 6, and if that fails Qt 5, under the name `QT`. If either of them is found, `find_package` will succeed, and the CMake variable `QT_VERSION_MAJOR` will be defined to either 5 or 6.

`find_package(<PackageName>...)` will succeed, and the CMake variable `QT_VERSION_MAJOR` will be defined to either 5 or 6.

We then do load the package for the determined Qt version again by creating the name on the fly. This is needed because `find_package` expects the package name to be either `Qt5` or `Qt6`, and will print an error otherwise.

We can use the same pattern to also specify the name of the imported library. Before calling `target_link_libraries`, CMake will resolve to either `Qt5::Core` or `Qt6::Core`.

`target_link_libraries(helloworld PRIVATE Qt${QT_VERSION_MAJOR}::Core)`

建议的做法

尽可能使用 CMake 命令的无版本变体。

Topics >

如果您需要支持早于 Qt 5.15 的 Qt 5 版本，或者如果无法控制 CMake 代码是在可能定义了 `QT_NO_CREATE_VERSIONLESS_FUNCTIONS` 或 `QT_NO_CREATE_VERSIONLESS_TARGETS` 的上下文中加载的，请使用 CMake 命令和目标的版本化版本。在这种情况下，您仍可以通过变量确定实际命令或目标名称来简化代码。

视窗中的统一码支持

在 Qt 6 中，默认情况下，为链接到 Qt 模块的目标设置和编译器定义。这与 qmake 行为一致，但与 Qt 5 中的 CMake API 行为相比，这是一个变化。

在 Qt 6 中，默认情况下，为链接到 Qt 模块的目标设置和编译器定义。这与 qmake 行为一致，但与 Qt 5 中的 CMake API 行为相比，这是一个变化。

```
find_package(Qt6 COMPONENTS Core)

add_executable(helloworld
    ...
)

qt_disable_unicode_defines(helloworld)
```

[◀ 导入的目标](#)

[命令参考 ▶](#)

©2022 Qt Ltd. 此处包含的文档贡献是其各自所有者的版权。此处提供的文档是根据自由软件基金会发布的 [GNU 自由文档许可证 1.3 版](#)的条款进行许可的。Qt及其相应的徽标是Qt有限公司在芬兰和/或全球其他国家的[商标](#)。所有其他商标均为其各自所有者的财产。



联系我们

公司

- 关于我们
- 投资者
- 编辑部
- 职业
- 办公地点

发牌

- 条款及细则
- 开源
- 常见问题

支持

- 支持服务
- 专业服务
- 合作伙伴
- 训练

对于客户

- 支持中心
- 下载
- 秦特登录
- 联系我们
- 客户成功案例

社区

- 为Qt做贡献
- 论坛



顶部

© 2022 Qt公司

[反馈](#) [登录](#)