

[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.

无版本命令

从Qt 5.15开始，Qt模块还提供了其命令的无版本变体。例如，您现在可以使用`qt_add_translation`来编译翻译文件，而不管您使用的是 Qt 5 还是 Qt 6。

混合 Qt 5 和 Qt 6

可能有一些项目需要在一个 CMake 上下文中同时加载 Qt 5 和 Qt 6（尽管不支持在一个库或可执行文件中混合 Qt 版本，所以要小心）。

在这样的设置中，无版本目标和命令将隐式引用通过找到的第一个 Qt 版本。在第一次调用之前设置 `QT_DEFAULT_MAJOR_VERSION` CMake 变量，以使版本显式。 `find_package`

支持较旧的 Qt 5 版本

如果你还需要支持早于 Qt 5.15 的 Qt 5 版本，你可以通过将当前版本存储在 CMake 变量中来实现：

```
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)
```

在这里，我们尝试找到第一个 Qt 6，如果 Qt 5 失败，则以名称。如果找到其中任何一个，则将成功，并且 CMake 变量将定义为或。

`find_package(<PackageName>...)QTfind_packageQT_VERSION_MAJOR56`

然后，我们通过动态创建名称来再次加载已确定的 Qt 版本的包。这是必需的，因为期望包名称为或，否则将打印错误。 `Qt${QT_VERSION_MAJOR}CMAKE_AUTOMOCQt5Qt6`

我们可以使用相同的模式来指定导入库的名称。在调用之前，CMake 将解析为或。

`target_link_librariesQt${QT_VERSION_MAJOR}::WidgetsQt5::WidgetsQt6::Widgets`

建议的做法

尽可能使用 CMake 命令的无版本变体。

无版本导入的目标对于需要同时使用 Qt 5 和 Qt 6 进行编译的项目非常有用。由于缺少目标属性，因此不建议默认使用它们。

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。 `UNICODE_UNICODE`

在目标上调用 `qt_disable_unicode_defines()` 以不设置定义。

```
find_package(Qt6 COMPONENTS Core)
```

```
)

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
- 维基
- 下载
- 市场



