平台说明

许多跨平台项目都可以通过基本的 qmake 配置功能进行处理。但是，在某些平台上，利用特定于平台的功能有时很有用，甚至是必要的。qmake知道其中的许多功能，这些功能可以通过特定的变量进行访问，这些变量仅在它们相关的平台上生效。

苹果操作系统、苹果系统、电视操作系统和手表操作系统

特定于这些平台的功能包括支持创建通用二进制文件、框架和捆绑包。

源代码包和二进制包

源包中提供的 qmake 版本的配置与二进制包中提供的版本略有不同，因为它使用不同的功能规范。如果源包通常使用规范，则二进制包通常配置为使用规范。`macx-g++macx-xcode`

每个软件包的用户都可以通过使用该选项调用 qmake 来覆盖此配置（有关更多信息，请参阅运行 qmake）。例如，要使用二进制包中的 qmake 在项目目录中创建 Makefile，请调用以下命令：`-spec`

```
qmake -spec macx-g++
```

使用框架

qmake 能够自动生成构建规则，用于针对 macOS 上标准框架目录中的框架进行链接，该目录位于。`/Library/Frameworks/`

需要将标准框架目录以外的目录指定给构建系统，这是通过将链接器选项附加到 LIBS 变量来实现的，如以下示例所示：

```
LIBS += -F/path/to/framework/directory/
```

通过将选项和框架名称附加到 LIBS 变量来链接框架本身：`-framework`

```
LIBS += -framework TheFramework
```

模板，并将该选项添加到 CONFIG 变量中：lib_bundle

```
TEMPLATE = lib
CONFIG += lib_bundle
```

与库关联的数据是使用 QMAKE_BUNDLE_DATA 变量指定的。它包含将与库捆绑包一起安装的项目，并且通常用于指定头文件的集合，如以下示例所示：

```
FRAMEWORK_HEADERS.version = Versions
FRAMEWORK_HEADERS.files = path/to/header_one.h path/to/header_two.h
FRAMEWORK_HEADERS.path = Headers
QMAKE_BUNDLE_DATA += FRAMEWORK_HEADERS
```

您可以使用该变量指定特定框架所需的标头。将其附加到变量可确保将有关这些标头的信息添加到将与库捆绑包一起安装的资源集合中。此外，框架名称和版本由QMAKE_FRAMEWORK_BUNDLE_NAME和QMAKE_FRAMEWORK_VERSION变量指定。默认情况下，用于这些变量的值是从"目标"和"版本"变量中获取的。FRAMEWORK_HEADERSQMAKE_BUNDLE_DATA

请参阅 qt for macOS - 部署 以获取有关部署应用程序和库的更多信息。

创建和移动 Xcode 项目

在 macOS 上，开发人员可以从现有的 qmake 项目文件生成 Xcode 项目文件。例如：

```
qmake -spec macx-xcode project.pro
```

注意： 如果项目稍后在磁盘上移动，则必须再次运行 qmake 以处理项目文件并创建新的 Xcode 项目文件。

同时支持两个构建目标

目前实现这一点并不可行，因为活动构建配置的 Xcode 概念在概念上与构建目标的 qmake 概念不同。

"Xcode 活动生成配置"设置用于修改 Xcode 配置、编译器标志和类似的生成选项。与 Visual Studio 不同，Xcode 不允许根据是否选择了调试或发布生成配置来选择特定的库文件。qmake 调试和发布设置控制将哪些库文件链接到可执行文件。

It is currently not possible to set files in Xcode configuration settings from the qmake generated Xcode project file. The way the libraries are linked in the *Frameworks & Libraries* phase in the Xcode build system.

Furthermore, the selected *Active Build Configuration* is stored in a .pbxuser file, which is generated by Xcode on first load, not created by qmake.

Windows

2005, or later.

Adding Windows Resource Files

This section describes how to handle a Windows resource file with qmake to have it linked to an application executable (EXE) or dynamic link library (DLL). qmake can optionally auto-generate a suitably filled Windows resource file.

A linked Windows resource file may contain many elements that can be accessed by its EXE or DLL. However, the Qt resource system should be used for accessing linked-in resources in a platform-independent way. But some standard elements of the linked Windows resource file are accessed by Windows itself. For example, in Windows explorer the version tab of the file properties is filled by resource elements. In addition, the program icon of the EXE is read from these elements. So it is good practice for a Qt created Windows EXE or DLL to use both techniques at the same time: link platform-independent resources via the Qt resource system and add Windows specific resources via a Windows resource file.

Typically, a resource-definition script (.rc file) is compiled to a Windows resource file. Within the Microsoft toolchain, the RC tool generates a .res file, which can be linked with the Microsoft linker to an EXE or DLL. The MinGW toolchain uses the windres tool to generate an .o file that can be linked with the MinGW linker to an EXE or DLL.

The optional auto-generation of a suitably filled .rc file by qmake is triggered by setting at least one of the system variables VERSION and RC_ICONS. The generated .rc file is automatically compiled and linked. Elements that are added to the .rc file are defined by the system variables QMAKE_TARGET_COMPANY, QMAKE_TARGET_DESCRIPTION, QMAKE_TARGET_COPYRIGHT, QMAKE_TARGET_PRODUCT, QMAKE_TARGET_ORIGINAL_FILENAME, QMAKE_TARGET_INTERNALNAME, QMAKE_TARGET_COMMENTS, QMAKE_TARGET_TRADEMARKS, QMAKE_MANIFEST, RC_CODEPAGE, RC_ICONS, RC_LANG and VERSION.

If these elements are not sufficient, qmake has the two system variables RC_FILE and RES_FILE that point directly to an externally created .rc or .res file. By setting one of these variables, the specified file is linked to the EXE or DLL.

> **Note:** The generation of the .rc file by qmake is blocked, if RC_FILE or RES_FILE is set. In this case, no further changes are made to the given .rc file or the .res or .o file by qmake; the variables pertaining to .rc file generation have no effect.

Creating Visual Studio Project Files

This section describes how to import an existing qmake project into Visual Studio. qmake is able to take a project file and create a Visual Studio project that contains all the necessary information required by the development environment. This is achieved by setting the qmake project template to either (for application projects) or (for library projects).`vcapp``vclib`

This can also be set using a command line option, for example:

```
qmake -tp vc
```

It is possible to recursively generate files in subdirectories and a file in the main directory, by typing:`.vcproj``.sln`

```
qmake -tp vc -r
```

Note: If you are using the Visual Studio Add-in, select **Qt** > **Import from .pro file** to import files. `.pro`

Visual Studio Manifest Files

When deploying Qt applications built using Visual Studio 2005, or later, make sure that the manifest file that was created when the application was linked is handled correctly. This is handled automatically for projects that generate DLLs.

Removing manifest embedding for application executables can be done with the following assignment to the CONFIG variable:

```
CONFIG -= embed_manifest_exe
```

Also, the manifest embedding for DLLs can be removed with the following assignment to the variable:`CONFIG`

```
CONFIG -= embed_manifest_dll
```

This is discussed in more detail in the deployment guide for Windows.

‹ Running qmake qmake Language ›

Qt DOCUMENTATION

Support Services
Professional Services
Partners
Training

Support Center
Downloads
Qt Login
Contact Us
Customer Success

Community

Contribute to Qt
Forum
Wiki
Downloads
Marketplace

Feedback Sign In