# qmake for MeVisLab - Documentation

# qmake for MeVisLab - Documentation

# Table of Contents

# Chapter 1. qmake and MeVisLab

## 1.1. Introduction to qmake

qmake is a makefile generator by Qt Software. It is a part of the Qt development platform but can also be used independently. It is available for many operating systems.

qmake "translates" the contents of the platform-independent project file .pro into a platform-dependent project file. For MeVisLab packages, the project file types resulting from a qmake run are

- vcxproj (Windows; currently supported for VC10, VC10-64 (Visual Studio 2010), VC11-64 (Visual Studio 2012), and VC12-64 (Visual Studio 2013); set in the environment variable MLAB_COMPILER_VERSION)
- xcodeproj (Mac)
- makefile (Linux)

Another similar, important file type are .pri files for the package configuration. They are linked in .pro files.

For a general manual and tutorial for qmake, please visit the manufacturer's website at Qt Software.

qmake is delivered with the MeVisLab SDK.

## 1.2. Syntax in .pro and .pri files

The full syntax can be found in the qmake Project Files Documentation.

An important part of the notation are scopes which are used to enter the platform-dependent settings. Scopes are similar to if statements in procedural programming languages. If a certain condition is true, the declarations inside the scope are processed.

A basic scope:

```
win32 {
           SOURCES += MLBase.cpp
      }
```

Scopes can be negated:

```
!win32 {
          SOURCES += MLBase.cpp
      }
```

Scopes can be nested like this:

```
win32 {
           contains(DEFINES,MEVIS_STLPORT) {
           DEFINES += BOOST_NO_STD_ALLOCATOR
           }
      }
```

...or nested like this:

```
macx:debug {
           HEADERS += debugging.h
      }
```

To perform single line conditional assignments, a : operator may be used, also for nested scopes:

```
    unix:LIBS += -lpthread
```

```
    unix:debug:LIBS += -lpthread
```

Important scopes are `win32`, `unix`, `linux`, `macx`, `debug` and `release`.

By Qt definition, functions are written lower case (`include`), variables are in capital letters (`CONFIG`).

## 1.3. .pro Files in MeVisLab

In general Qt context, a .pro (project) file includes the necessary details so that a platform-dependent makefile can be created by qmake.

In MeVisLab, the .pro files largely adhere to the Qt .pro standard. However, some entries are MeVisLab-specific, especially the MLAB_PACKAGES entry.

A typical example (excerpt from SoUtils.pro):

```
MEVIS_MAINTAINER = schumann

TEMPLATE = lib               # Qt setting for generation

TARGET  = SoUtils            # explicit target name

DESTDIR    = ../../../lib    # destination for lib files
DLLDESTDIR = ../../../lib    # destination for dll files

WARN=MAXIMUM64               # compiler warn levels

# add dependencies of this project to other projects here
CONFIG += dll MLOpenGL ftgl inventor MLInventorBinding MLLUT MLUtilities ML

MLAB_PACKAGES += MeVisLab_Standard  # packages needed due to CONFIG dependencies

# make sure that this file is included after CONFIG and MLAB_PACKAGES
include ($(MLAB_MeVis_Foundation)/Configuration/IncludePackages.pri)

DEFINES += SOUTILS_EXPORTS  # adds the value as compiler C++ preprocessor macros

HEADERS += \
    SoBackground.h \
    SoCalculatorWrapper.h \
    SoClipBox.h

SOURCES += \
    SoBackground.cpp \
    SoCalculatorWrapper.cpp \
    SoClipBox.cpp

# additional files that are NOT compiled
RELATEDFILES += \
../../../Modules/Inventor/SoUtils/utils.def
```

## 1.4. .pri Files in MeVisLab

In general Qt context, a .pri file contains the list of sources, headers,.ui and .qrc files in the project.

In MeVisLab, the .pri file is the package configuration file. It lists all projects that belong to the package. The .pri file is the key to resolve the dependencies that are given in the CONFIG and

MLAB_PACKAGES entries of the .pro files. Each package contains one .pri file, which by convention is named [PackageGroup]_[Package].pri.

The variables given in the .pri and .pro files are set as environment variables by scripts before qmake is started.

For the reader's convenience, a .pri file typically starts with the list of included modules, followed by the possible inclusion of other .pri files and the definition of PACKAGE_ROOT and PACKAGE_SOURCES. After this, the modules and their configuration follow.

`CONFIG_FOUND = [ProjectName]` is used to check whether the project is actually existing after scanning the CONFIG += line in a .pro-file.

A typical example (excerpt from MeVisLab_Standard.pri):

```
isEmpty(MEVISLAB_STANDARD_PRI_INCLUDED) {
  message ( loading MeVisLab_Standard.pri )
}

...

# Standard Packages:
# Packages available in the standard MeVisLab SDK

# -- ML and ML Modules
# MLBackgroundTasks      - Support for multithreaded background tasks
# MLBase                 - Basic ListTypes etc. (Tobias, Florian)
# MLSmallImageInterface  - Interface for setting up first steps, place before MLTools!

...

# include guard against multiple inclusion
isEmpty(MEVISLAB_STANDARD_PRI_INCLUDED) {

MEVISLAB_STANDARD_PRI_INCLUDED = 1

# -- System -------------------------------------------------------------

include( $(MLAB_MeVis_BuildSystem)/Configuration/SystemInit.pri )

# -- Define local PACKAGE variables ------------------------------------

PACKAGE_ROOT    = $$(MLAB_MeVisLab_Standard)
PACKAGE_SOURCES = "$$(MLAB_MeVisLab_Standard)"/Sources

# Add package library path
LIBS          += -L"$${PACKAGE_ROOT}"/lib

# -- ML -------------------------------------------------------------

MLBackgroundTasks {
  CONFIG_FOUND += MLBackgroundTasks
  INCLUDEPATH += $${PACKAGE_SOURCES}/ML/MLBackgroundTasks
  win32:LIBS += MLBackgroundTasks$${d}.lib
  unix:LIBS += -lMLBackgroundTasks$${d}
  macx:CONFIG += ML
}

MLBinaryTree {
CONFIG_FOUND += MLBinaryTree
INCLUDEPATH += $${PACKAGE_SOURCES}/ML/MLBinaryTree
win32:LIBS += MLBinaryTree$${d}.lib
unix:LIBS += -lMLBinaryTree$${d}
```

```
  macx:CONFIG += ML
  }
 ...

  # End of packages --------------------------------------------------

  } # END MEVISLABPRO_STANDARD_PRI_INCLUDED
```

# Chapter 2. qmake for MeVisLab Reference

## 2.1. MEVIS_MAINTAINER (MeVisLab-specific variable)

Optional: Gives the name of the MeVisLab developer maintaining this package.

Example:

```
    MEVIS_MAINTAINER = Florian.Link
```

The name is linked to an internal e-mail address so that in case of events like errors in the build of the module, the maintainer automatically receives an e-mail notification.

## 2.2. TEMPLATE

Mandatory: Enter the name of the template to be used when generating the project (as defined by Qt).

The two important types for MeVisLab:

```
    TEMPLATE = app    # create stand-alone application executable
```

```
    TEMPLATE = lib    # create a library (standard)
```

For the library types (static, dll etc.), an entry needs to be added to CONFIG.

## 2.3. TARGET

Mandatory: Substitute the normal Qt target naming based on the .pro file name with your own target name.

Example:

```
    TARGET = MLABControls
```

Usually the TARGET is named after the project file. However, as the TARGET value is read by a script in MeVisLab, it has to be entered explicitly. We recommend that you use the project name as target, like in the convention.

## 2.4. Directory settings

Optional: Enter destination directories for the target file (lib) and for a copy of the target dll (bin or lib). The paths are relative to the repository.

Example for lib (standard):

```
    DESTDIR       = ../../../lib
    DLLDESTDIR   = ../../../lib
```

Example for app (e.g. in MeVisLab IDE):

```
    DESTDIR       = ../../../bin
```

The path depth depends on your package structure. Typically, .pro files should put the results into the lib/bin directory of the package their are located in.

## 2.5. WARN (MeVisLab-specific variable)

Optional: Set warning levels. This variable has to be entered before including the .pri file.

Examples:

```
    # Set high warn level (warn 4 on MSCV)
    WARN = HIGH
```

```
    # Set maximum warn level with checks for 64 bit problems
    WARN = MAXIMUM64
```

## 2.6. CONFIG

Mandatory: Add the used projects here. Note that a .pro file can include more than one CONFIG line.

In a MeVisLab context, the CONFIG entries give the dependencies of the current project to other projects. These dependencies are then resolved via the .pri files of the packages given in MLAB_PACKAGES. Example:

```
    CONFIG += MLABIDE MLABBase
```

In a general Qt context, CONFIG variables also define the application/library type. Examples:

```
     linux:CONFIG += x11   # target is a X11 application or library
     CONFIG += dll         # target is a dll
     CONFIG += staticlib   # target is static library (lib only)
     CONFIG += qt          # target is Qt application
```

The resulting CONFIG lines may differ depending on the module type/background.

Example for an ML module:

```
    CONFIG += dll ML MLTools
```

Example for an OpenInventor module:

```
    CONFIG += dll inventor MLOpenGL MLInventorBinding ML
```

Example for an itk-based module:

```
    CONFIG += dll itk
```

Example for a vtk-based module:

```
    CONFIG += dll vtk MLVTKSupport ML
```

For a module that relates to several other modules, the resulting CONFIG may be longer than a line. Example from MLImageFile.pro:

```
    CONFIG += dll MLUtilities MLImageIO MLImageIO_DICOM ML \
              MLDicomTree_OFFIS MLDicomTreeImagePropertyExtension
```

For details, see CONFIG in the qmake Reference.

## 2.7. MLAB_PACKAGES (MeVisLab-specific variable)

Mandatory: Gives the package(s) that this project depends on (for resolving the CONFIG line), like

- MeVisLab_IDE
- MeVisLab_Standard
- FMEwork_Release
- ...and others, including your user-specific package (groups).

Example:

A CONFIG entry like...

```
CONFIG += dll ML MLBase MLTools MLVolumetry
```

...would result in an MLAB_PACKAGES entry of:

```
MLAB_PACKAGES += MeVisLab_Standard \
                 FMEwork_Release \
                 FMEstable_General
```

I.e. all packages that are needed to resolve the dependencies given in CONFIG.

## 2.8. include

Mandatory: Include a .pri file in the .pro file.

In a general Qt context, the `include (filename)` function includes the contents of the specified file. The included file is processed immediately.

In the MeVisLab context, the include function is used specifically to include the .pri file which is needed to resolve the dependencies to other packages and projects. It has to be entered after the entries for CONFIG and MLAB_PACKAGES as those lines set the variables for which the paths of the .pri file are processed.

Example:

```
# after CONFIG and MLAB_PACKAGES
include ($(MLAB_MeVis_Foundation)/Configuration/IncludePackages.pri)
```

For more information, see the qmake Function Reference.

## 2.9. DEFINES

Optional: Add the values of this variable as compiler C++ preprocessor macros (-D option).

Example:

```
DEFINES += MEVISLAB_CORE MEVISLAB_CONTROLS_EXPORTS UNICODE
```

These DEFINES may be used for IFDEF or IFNDEF statements, for example in mlabBasics.cpp:

```
namespace MLABBase {
   MEVISLAB_BASE_EXPORT void init()
  {
MLABLogging::init();
     ...
     #ifdef MEVISLAB_CORE
         MLABDiagnosis::init();
         MLABTreeValueValidatorFactory::init();
         ...
   #endif
}
```

## 2.10. Additional libraries

Optional: Add additional libraries to be used by qmake. These may be platform-dependent.

Examples:

```
# Additional libraries
win32:LIBS   += vfw32.lib shell32.lib
linus:LIBS += -lpthread
macx:LIBS += -framework ApplicationServices
unix:!macx:LIBS += -lglut -lX11
```

## 2.11. INCLUDEPATH

Optional: Add further includes and libraries. Placeholders can be used.

### Note

It is not recommended to add INCLUDEPATH to a .pro file. If possible, it should be added to the .pri file of the package instead.

Example:

```
# add additional includes and libraries here
INCLUDEPATH += SoftLib
```

## 2.12. Headers and Sources

Mandatory: Add the C++ headers and sources of the project files.

Example:

```
# Add your project headers here
HEADERS += \
    TransferFunctionBase.h \
    mlGradation.h \
    mlParamLut2D.h
```

```
# Add your project cpp files here
SOURCES += \
     TransferFunctionBase.cpp \
     mlGradation.cpp \
     mlParamLut2D.cpp
```

## 2.13. Additional files

Optional: Add additional (related) files that are not compiled. The files may be platform-dependent.

Example:

```
# Additional files that are NOT compiled
    RELATEDFILES += \
    MLFouCollections.bat \
    ../../../Modules/ML/MLFourier/MLFourier.def \
    ../../../Modules/ML/MLFourier/mlFFT2d.def
```

Absolute paths may be necessary under Windows to link to network drives. However, this kind of linking should be avoided.