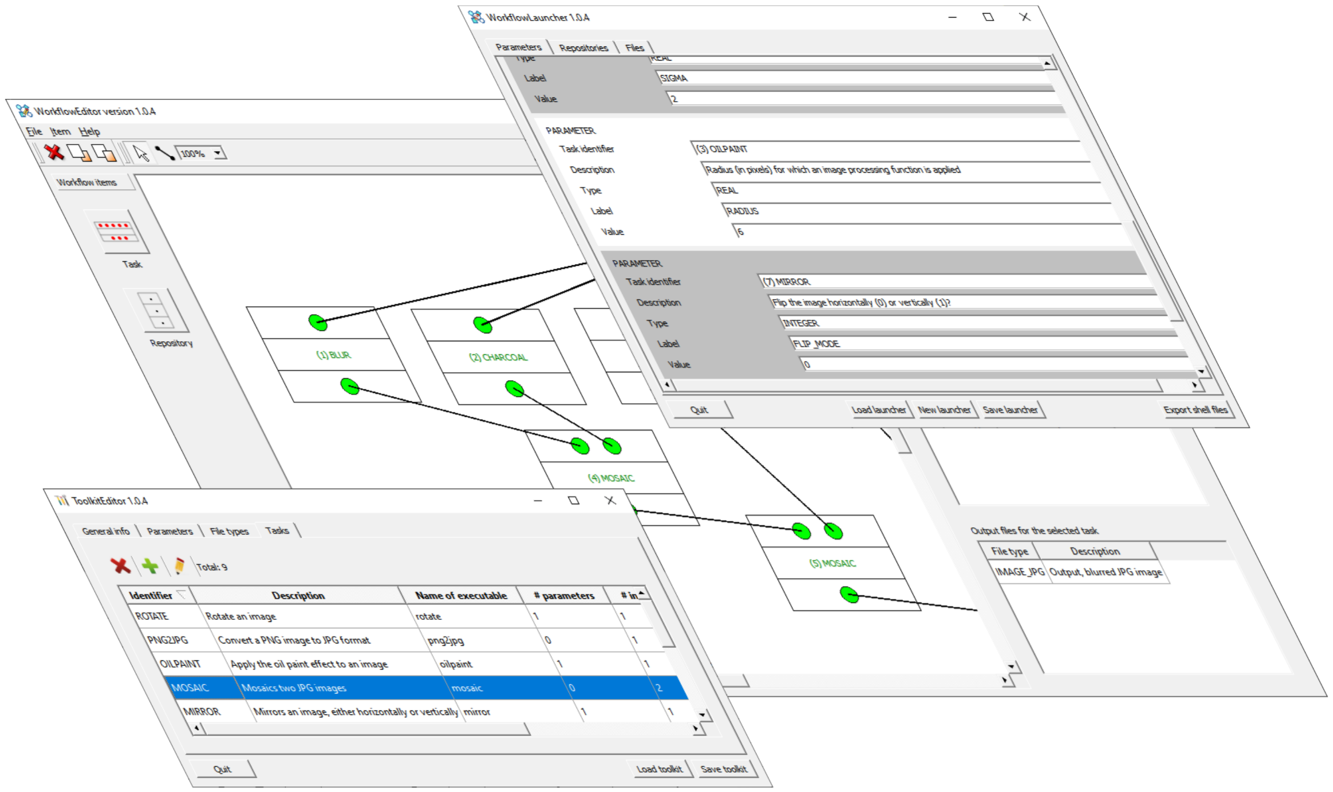
Building WorkflowMaker

For WorkflowMaker version **1.0.5** and later
Windows 10 & 11, Ubuntu-based Linux distributions
64-bit only



Label	SIGMA
Value	2

PARAMETER

Task identifier	(3) OILPAINT
Description	Radius (in pixels) for which an image processing function is applied
Type	REAL
Label	RADIUS
Value	6

PARAMETER

Task identifier	(7) MIRROR
Description	Flip the image horizontally (0) or vertically (1)?
Type	INTEGER
Label	FLIP_MODE
Value	0

File type	Description
IMAGE_JPG	Output, blurred JPG image

Identifier	Description	Name of executable	# parameters	# in
ROTATE	Rotate an image	rotate	1	1
PNG2JPG	Convert a PNG image to JPG format	png2jpg	0	1
OILPAINT	Apply the oil paint effect to an image	oilpaint	1	1
MOSAIC	Mosaics two JPG images	mosaic	0	2
MIRROR	Mirrors an image, either horizontally or vertically	mirror	1	1

Table of contents

1 The tools....3
2 The source code....3
3 Building WorkFlowMaker....4
3.1 The postbuild.bat or postbuild.sh files....4
3.2 Building WorkFlowMakerLib....6
3.3 Building the rest of the tools....7
3.4 Building the API documentation....7

List of figures

Figure 1: The organization of the source code....4
Figure 2: The structure of the QT_redistributables folder....6

1 The tools

To build WorkFlowMaker from source the following tools are required:

- A C++ compiler. The compilers used to build the binary distributions were MSVC 2022 (Windows) and gcc (Linux). The instructions provided by this document will assume that these compilers will be used. This is very important in the case of Qt (see below). If other compilers are used the appropriate version of Qt must be downloaded. If the compiler selected is not supported by pre-built versions of Qt, then the source code version of this tool will have to be downloaded and built manually from source.
- Qt (https://www.qt.io/download-open-source) version 5. See the comments above.
- doxygen (to build the API documentation).

Then,

- Download and install MSVC 2022.
- Download and install Qt 5. Remember to download the pre-built version of Qt matching the MSVC compiler.
- Download and install doxygen only if the API documentation must be generated. This documents are useful only if you want to modify the tool; it is irrelevant in any other circumstances.

2 The source code

The source code is organized as depicted in Figure 1 on page 4.

Each of the following subfolders:

- WorkFlowMakerLib,
- ToolKitEditor,
- WorkFlowEditor, and
- WorkFlowLauncher

contains, among others, two files, with extensions “pro” and “dox”.

The pro file is the project file for QtCreator, the code editor and interactive development environment from Qt. The dox file is the project file for doxygen.

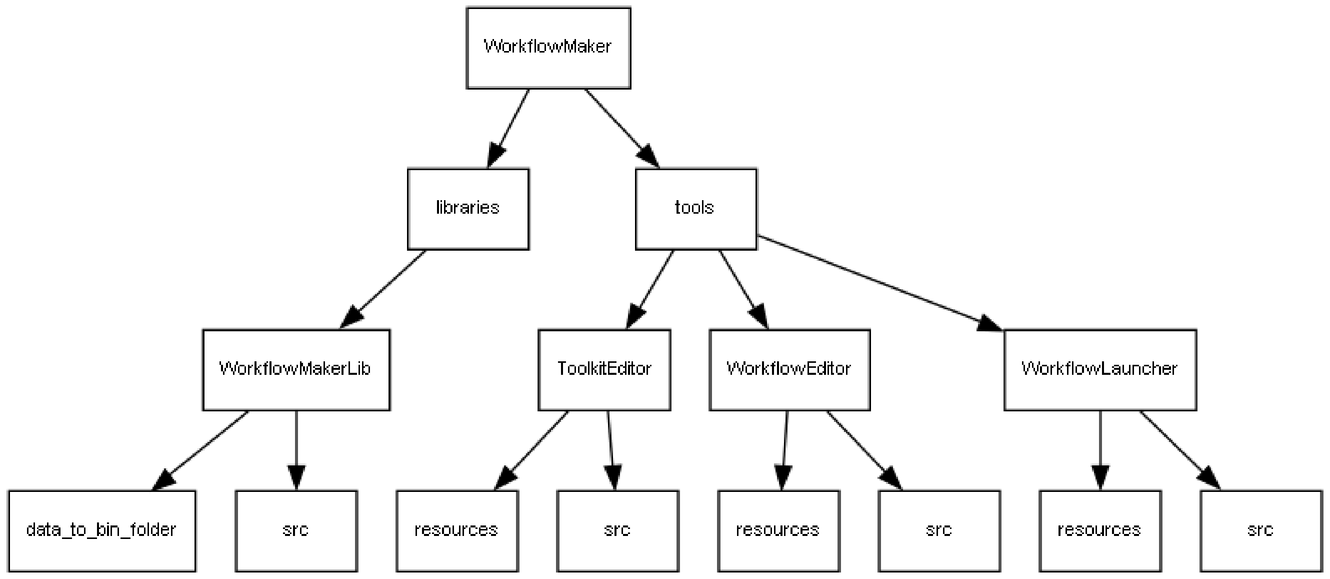


Figure 1: The organization of the source code.

3 Building WorkFlowMaker

In general, to build the code, follow the steps below:

- Open QtCreator.
- Select the pro file for the desired target (the library or any of the tools).
- Configure the project (selecting the Qt 5 installed in your computer and the target output).
- Build.

However, the build order is important. WorkFlowMakerLib must be built first. Then, the tools may be built in any order.

3.1 The postbuild.bat or postbuild.sh files

All the components (libraries and tools) include two files, namely postbuild.bat and postbuild.sh, targeted at performing a very specific task: copying some files to some folder to later facilitate the construction of installers. The bat files are for Windows systems, while the sh ones are required when building on a Linux platform.

If you are **not** interested in building an installer, you may omit the calls to the respective post build files modifying the pro files **(only for the tools)**; typically, these include a set of sentences like these:

```
# Post build step ONLY FOR RELEASE platforms (not for DEBUG ones.)

CONFIG(release, debug|release) {
  win32 {
    QMAKE_POST_LINK += $${PWD}/postbuild.bat $${PWD} $${PLATFORM} $${CURRENT_CONFIG} $${TARGET}
  }
  unix:!macx {
    QMAKE_POST_LINK += $${PWD}/postbuild.sh $${PWD} $${CURRENT_CONFIG} $${TARGET}
  }
}
```

After removing these sentences from the pro files, QtCreator will not call the post build files, thus avoiding any copy operation.

Assuming that you are interested in building an installer, then you should read very carefully the rest of this section. Otherwise, you may skip it completely.

Adhering to the following instructions it will be possible to guarantee that the post build files work properly. Not doing so implies that these will fail.

The kind of files that these post build files copy are:

- The executable (or library) generated by Qt.
- The file defining the version of the library (only when building the library).
- Sample files (examples).
- Schema XML files (to validate XML files, either toolkits, workflows, or launchers).
- The Qt runtime (only on Windows) and
- the Visual C++ runtime (only on Windows).

Prior to copying these files, the post build files create a folder named “installer”. Depending on the platform (Windows / Linux) the contents (subfolders) created for installer will vary. For instance, a subfolder for the binaries (executables, dynamic shared libraries) will be created on the Windows and Linux platforms; others for user guides or samples will be created on the Windows platform only. Depending on the type of file, it will be copied by the post build files to one or another of these subfolders.

On Windows (see the list above) the runtimes for both Qt and MSVC are copied, so these must be available somewhere (these are not provided in the GitHub repository). The post build files expect to find these runtimes in the folders named QT_redistributables and VC_redistributables respectively.

The QT_redistributables folder must contain all the Qt dynamic link libraries and platform code required by the applications. These are:

- Qt5Core.dll
- Qt5Gui.dll
- Qt5Widgets.dll
- Qt5XmlPatterns.dll

And the contents of the platforms subfolder must be (at least) this:

- qwindows.dll

The structure of the QT_redistributables folders must be the following one:

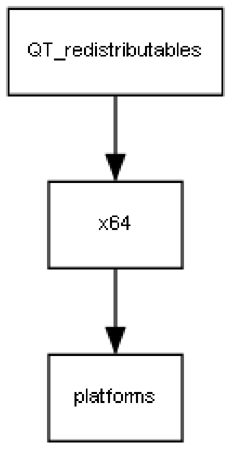


Figure 2: The structure of the QT_redistributables folder.

The Qt5*.dll files must be stored in the x64 subfolder. The qwindows.dll, in the platforms folder.

This is the structure and contents of the Qt_redistributables folder that the post build files expect to find. Then, these will be able to copy this structure to the bin subfolder of the installer folder.

Regarding the VC_redistributables folder, the most appropriate way to proceed is to copy the Microsoft's installer for the redistributables for the version of the compiler used to compile WorkFlowMaker in the VC_redistributables subfolder. The post build files will copy this installer to the binaries subfolder in the installer folder.

The idea of the post build files is to build a folder structure (installer) where all the files required to build an installer are copied, thus facilitating this task.

3.2 Building WorkFlowMakerLib

The first component to build must always be the WorkFlowMakerLib, since the tools (ToolkitEditor, WorkflowEditor and WorkflowLauncher) need this library to be correctly build.

When configuring the Qt project, it is possible to do it for both Release and Debug configurations. However, the project files for the tools are prepared to build Release configurations only, so it is your choice whether to ask for a Debug configuration for the library.

Concerning the post build file, it performs the following tasks:

1. It creates two new subfolders, namely “binaries” and “headers” just under the WorkFlowMakerLib folder (see Figure 1). When the library is built, a copy of the .h/.hpp files it contains are copied to the subfolder headers. The library itself is copied to the subfolder binaries.
2. It creates the installer folder (as described in the previous section).
3. It copies the contents of the subfolder data_to_bin_folder (see, again, Figure 1) to one of the subfolders of the newly created installed folder.

Step 1 above **should never be removed** from the post build files (see section 3.1) even in the case that you are not interested in building an installer. If this step is omitted, the process of building the tools will fail, since their pro files state that the headers / binaries of the library will be found in these subfolders (headers / binaries).

Step 2 may be removed (deleted from the post build files) if no installer will be generated.

However, step 3 is important. The unique file included in the data_to_bin_folder, named workflowmaker_version.txt is used by all the tools to find what is the version of the package. These tools look for this file in the folder where their executable file resides. If it is not found, they display “Unknown version” instead of the version of the software. Therefore, and if step 3 is omitted from the post build files, this file (workflowmaker_version.txt) must be copied in the applications’ folder.

Then to build the library, follow the steps indicated at the beginning of section 3.

3.3 Building the rest of the tools

Prior to building the tools, consider whether an installer will be built or not. Please read section 3.1 carefully to learn how to proceed in both cases.

Having built the WorkFlowMaker library, follow the steps at the beginning of section 3 to build, one by one, the three tools.

Consider, however, that the pro files for the tools are prepared to build Release configurations only; therefore, when configuring each project, select only the Release configuration, since trying to build the Debug one will not work.

3.4 Building the API documentation

To build the API documentation of the API of the library and the tools, run doxygen using the dox file present in each subfolder (WorkFlowMakerLib, ToolKitEditor, WorkflowEditor, WorkflowLauncher, see Figure 1). This will generate a subfolder named html. Open the index.html file inside the html subfolder to see the documentation.