Video Metadata Framework

Building guide – version 3.0

Revision History

Revision	Description	Date	Author
1.0	Initial revision	08 Dec 2013	Ivan Gubochkin, Dmitry Bogdanov, Andrew Senin
2.0	Extracted from User Guide	10 Nov 2014	Anna Plyakina
3.0	Updated for version 3.0	25 May 2016	Erik Niemeyer

List of content

Introduction

Video Metadata Framework (VMF) provides functionality for creating, editing and embedding of metadata into video files.

This User Guide gives a quick overview to the library installation package, its folder structure, describes the samples and demo application and explains how to build the library and use it in user applications.

For functionality overview and introduction into development with the library please review the specification document (VMF-Specification.pdf).

For architecture overview and low level implementation details please read the Architecture Design Document (VMF_Architecture.pdf, this document is not part of the standard installer).

For detailed information of the library functions and their parameters please use the Doxygen documentation shipped with the library (API Specification).

Building instructions

Building VMF from source code

In case your distribution contains source code of the library the following sections describe how to compile the library on different platforms.

Building under Windows

Dependencies

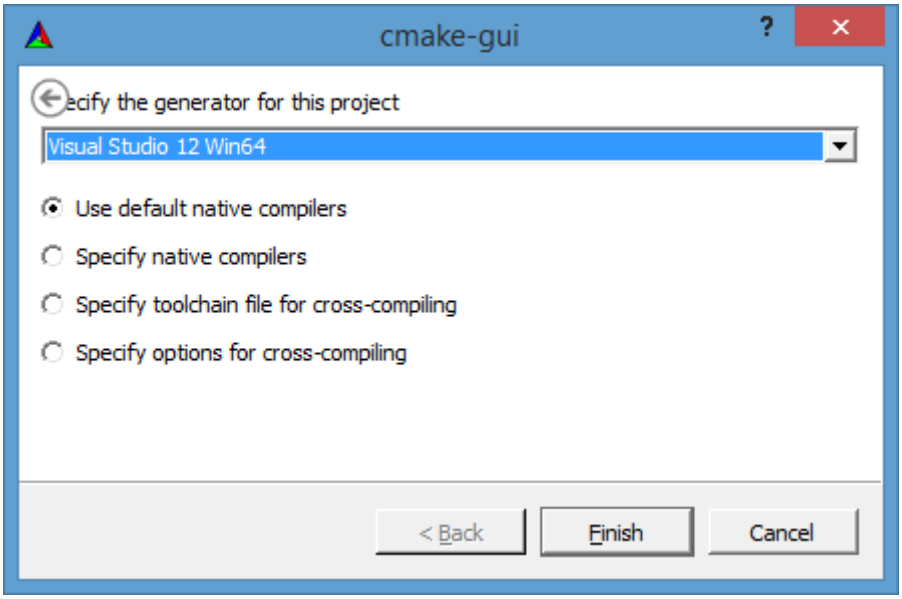
- Install MS VS 2013 (any addition) or later from https://www.visualstudio.com
- Install CMake 2.8.12 or higher (3.x is recommended) from http://www.cmake.org/cmake/resources/software.html
- [optionally] Install Doxygen 1.8.2 or higher from http://ftp.stack.nl/pub/users/dimitri/doxygen-1.8.2-setup.exe (if you'd like to generate VMF API reference)
- [optionally] Install Graphviz 2.34 or higher from http://www.graphviz.org/Download_windows.php and add path to the bin folder to the PATH system variable (if you'd like to generate graph items for VMF API reference)
- [optionally] Install Qt 5 from http://qt-project.org/downloads (if you'd like to build Qt-based samples)
- [optionally] Install OpenCV 2.4.6 or later from http://sourceforge.net/projects/opencvlibrary/files/opencv-win/ (if you'd like to build OpenCV-based samples)
- [optionally] Install Android NDK from http://developer.android.com/tools/sdk/ndk/index.html (if you'd like to build lib-s for Android)
- [optionally] Install WiX toolset 3.8+ from http://wixtoolset.org/ (if you'd like to build MSI installer package)
- Reboot your system to make sure all environment variables are registered after installation of the software packages

Building

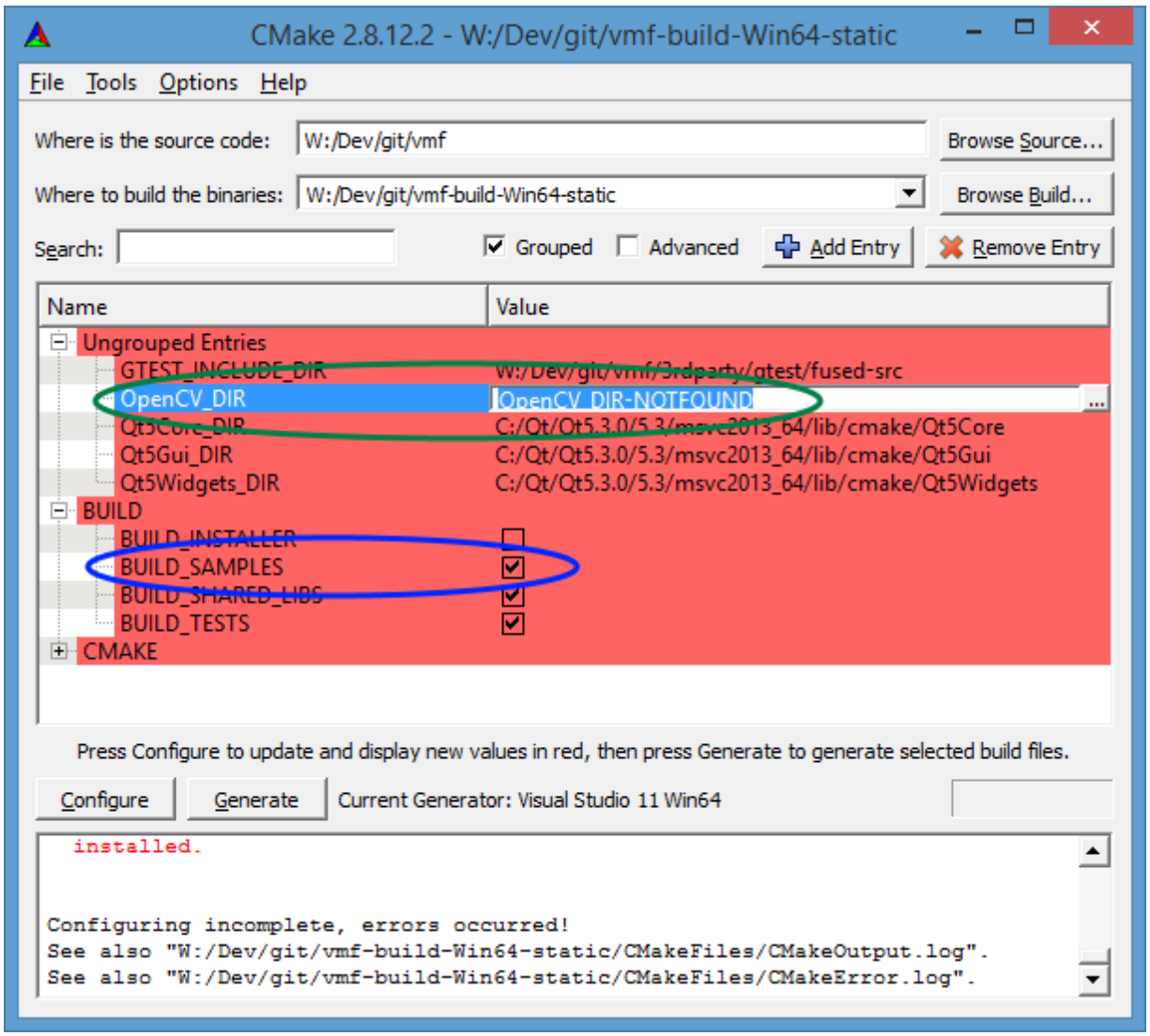
- Use console CMake within the build folder with the following options to configure the build:
  cmake.exe –G "Visual Studio 12 Win64" -DBUILD_TESTS=ON -DBUILD_SAMPLES=ON -DBUILD_QT_SAMPLES=OFF -DBUILD_SHARED_LIBS=ON path/to/src/dir
  (change the option values as your needs)
- Use console CMake within the build folder with the following options to perform the build:
  cmake.exe --build . --target INSTALL --config Release

or

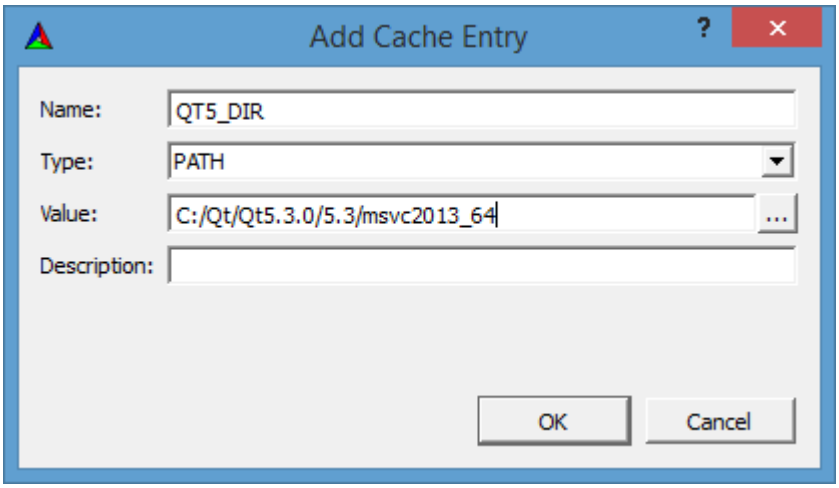
- Open CMake GUI
- In the text field “Where is the source code” enter full path to the source tree
- In the text field “Where to build the binaries” enter full path to the build tree (it is recommended to use build folder outside <SOURCES DIR>)
- Press “Configure” button
- Choose Visual Studio version which will be used for compilation (2013 and 2015 are supported as of VMF v3.0 release)



- Set OpenCV_DIR variable to a correct path to OpenCV build folder. Press “Configure” again after adding or changing the variable value.
  Alternatively you can uncheck BUILD_QT_SAMPLES checkbox since OpenCV is needed for those amples only



- If CMake can't find Qt5 you can either uncheck BUILD_QT_SAMPLES (since Qt5 is also needed for those samples only) or press "Add Entry" and a variable named QT5_DIR of type PATH keeping the path to your Qt5 binary directory. Then press "Configure" again.



- If configuring completes successful (“Configuring done” line appears in the lower log pane) - press “Generate” button
- In case all settings are correct (“Generating done” line appears in the lower log pane) CMake creates a Visual Studio solution at the folder set in the “Where to build the binaries” text field
- Open the generated vmf solution (vmf.sln) in Visual Studio IDE
- Select Debug or Release configuration and build the INSTALL target
- Please make sure you have Qt and OpenCV dll folders in your PATH variable so that system can load the dlls automatically when you start the VMF samples. Alternatively you can copy the OpenCV and Qt dlls to the folder with the executable

Building under Linux

Dependencies

- Ensure you have c++ compiler installed (i.e. gcc and g++ packets of version 4.7+ or clang and clang++ of version 3.2+ are installed)
- Install CMake 2.8.12 or higher (3.x is recommended) (http://www.cmake.org/cmake/resources/software.html)
- [optionally] Install Doxygen 1.8.2 (http://www.stack.nl/~dimitri/doxygen/download.html)
- [optionally] Install Graphviz 2.34 (http://www.graphviz.org/Download..php)
- [optionally] Build OpenCV 2.4.6+ (https://github.com/Itseez/opencv/releases) according to the “Building OpenCV from Source Using CMake, Using the Command Line” section of the “Installation in Linux” instruction (http://docs.opencv.org/doc/tutorials/introduction/linux_install/linux_install.html)
- [optionally] Install Qt 5: download (http://qt-project.org/downloads) and build according to the INSTALL file inside of the archive

Building

- Unpack the VMF source code to the location of your choice
- Change your current directory containing the source code tree and create ‘build’ folder next to the source one
- Change your working directory to the location to ‘build’ dir
- Execute cmake -DOpenCV_DIR=path/to/OpenCV/build/folder -DQT5_DIR=path/to/Qt5/binaries –G'Unix Makefiles' path/to/VMF/sources

- Execute cmake -L <path to VMF sources> to display cmake configuration variables
- Execute 'make install'.
  Build results will be placed to directory build/install/

Building for Android

Dependencies

- Install CMake 2.8.12 or higher (http://www.cmake.org/cmake/resources/software.html)
- Install Android NDK (http://developer.android.com/tools/sdk/ndk/index.html)
- [optionally] Install Java Dev Kit (JDK)
  (http://www.oracle.com/technetwork/java/javase/downloads/index.html)

Building

- Set environment variable ANDROID_NDK value of path to NDK root
- In the directory with unpacked VMF sources run compile_for_android.bat on Windows or compile_for_android on Linux. Note, this step removes directory <path to unpacked VMF sources>/build
- Built result files will be placed in <path to unpacked VMF sources>/build/bin and <path to unpacked VMF sources>/build/lib
- For executing unit tests you should run Android SDK. Then choose Window->Android Virtual Device Manager, create device if needed, and start it. In order to copy needed files to virtual device you should use adb push command from Android SDK. Also you may login to device with help of adb shell. Note, that you need to create the same directories structure (vmf/build/bin and vmf/data/) on virtual device to protect test from unnecessary crashing

Building 'vmf.framework' for iOS.

Run Python script `build_vmf_ios.py` (vmf/platform/ios/build_vmf_framework.py) under Mac OS in terminal(console).
Command format:

.../vmf/platform/ios/build_vmf_framework.py <abs_path_to_output_dir>

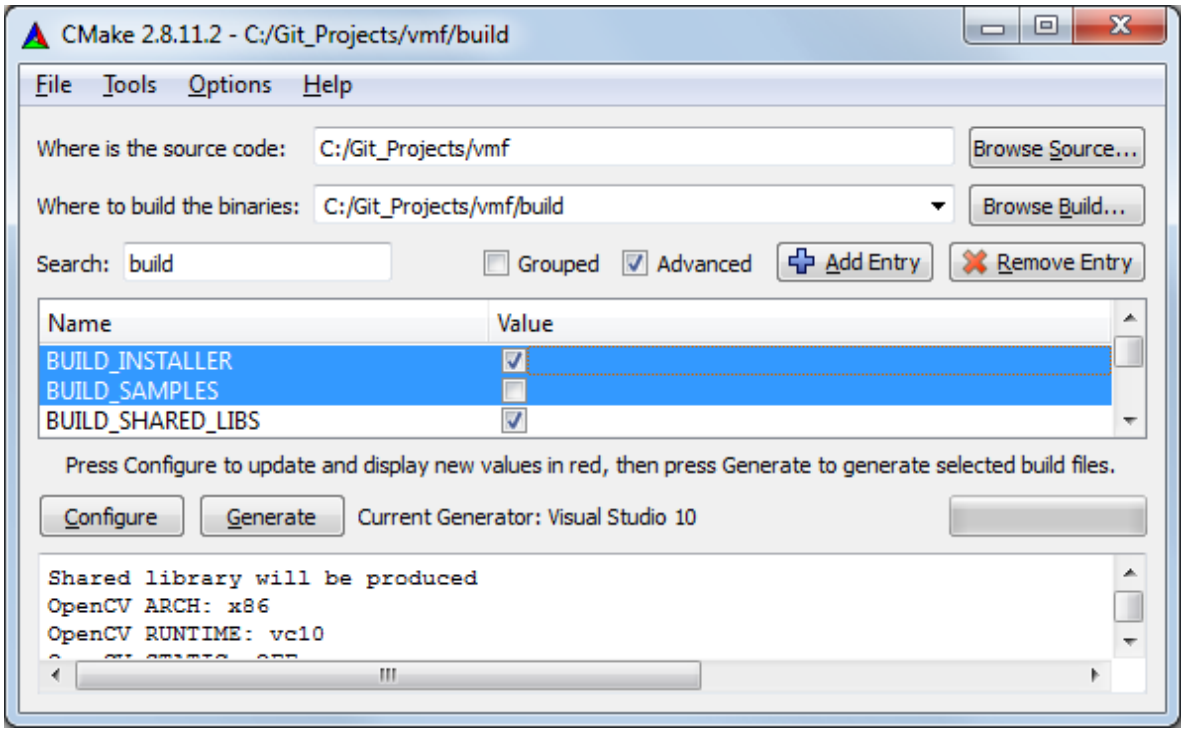
Creating Windows installer

Please see the “Dependencies” subsection of the “Building under Windows” section for the full list of tools necessary to build the installer.

Please follow the following steps to create the installer:

1. Configure CMake and generate Visual Studio solution by following the “Building under Windows” section

2. In CMake GUI turn on BUILD_INSTALLER flag and turn off BUILD_SAMPLES flag



3. Press the “Generate” button

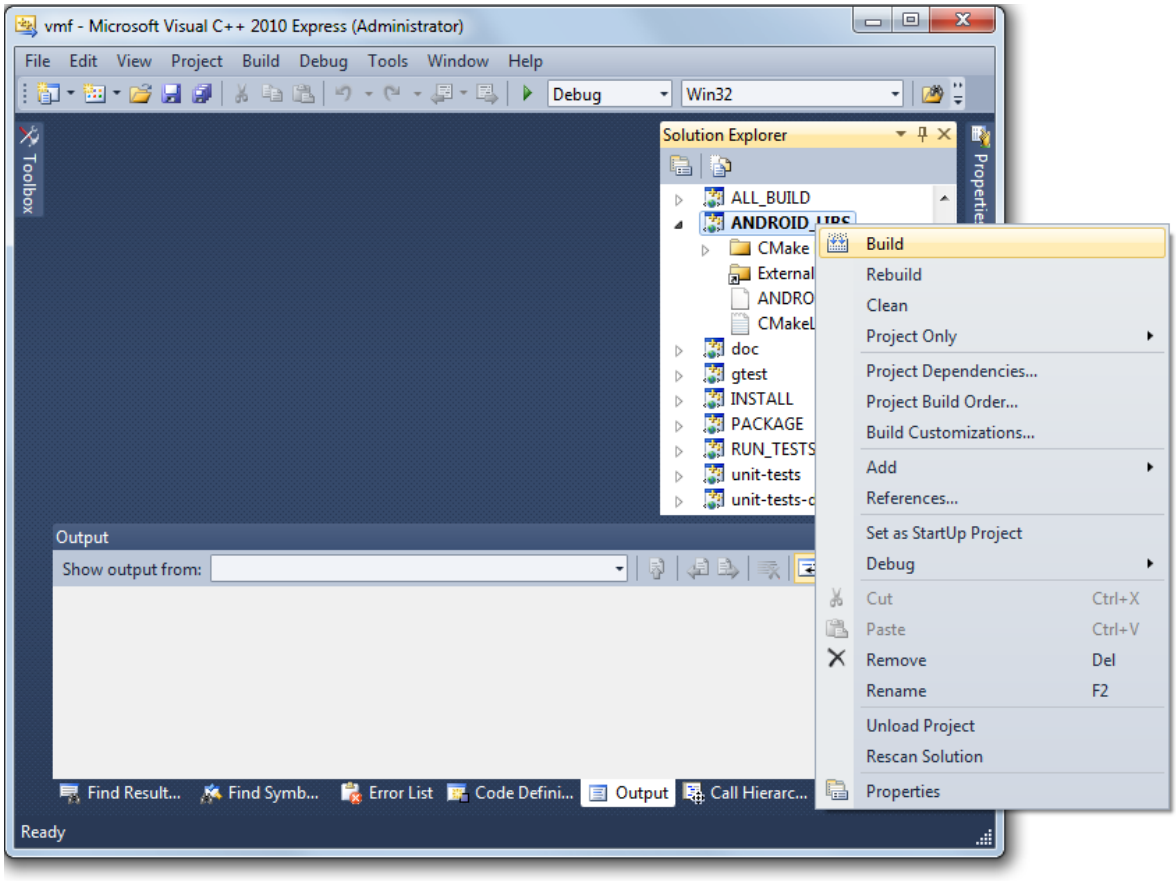
4. Make sure you have WiX toolset v3.8+ in your system and WIX environment variable keeps path to it (the path “%WIX%\bin\candle.exe” should exist).

5. Open vmf.sln from the folder set in the “Where to build the binaries” field in Visual Studio and build vmf project in Release and Debug configurations (“Build”->”Build Solution” menu item)

6. [Note] If you’d like to include both Release and Debug binaries to the installation package – build ‘INSTALL’ target for one of them (say Debug) then close MSVS, create “to_be_installed” folder in the top-level VMF **sources** folder, copy “x86” or “x64” folder from “<build-dir>/install” to “<source-dir>/to_be_installed” and re-run CMake.

(These are the same steps that are required for packaging several MSVS versions builds to a single package.)

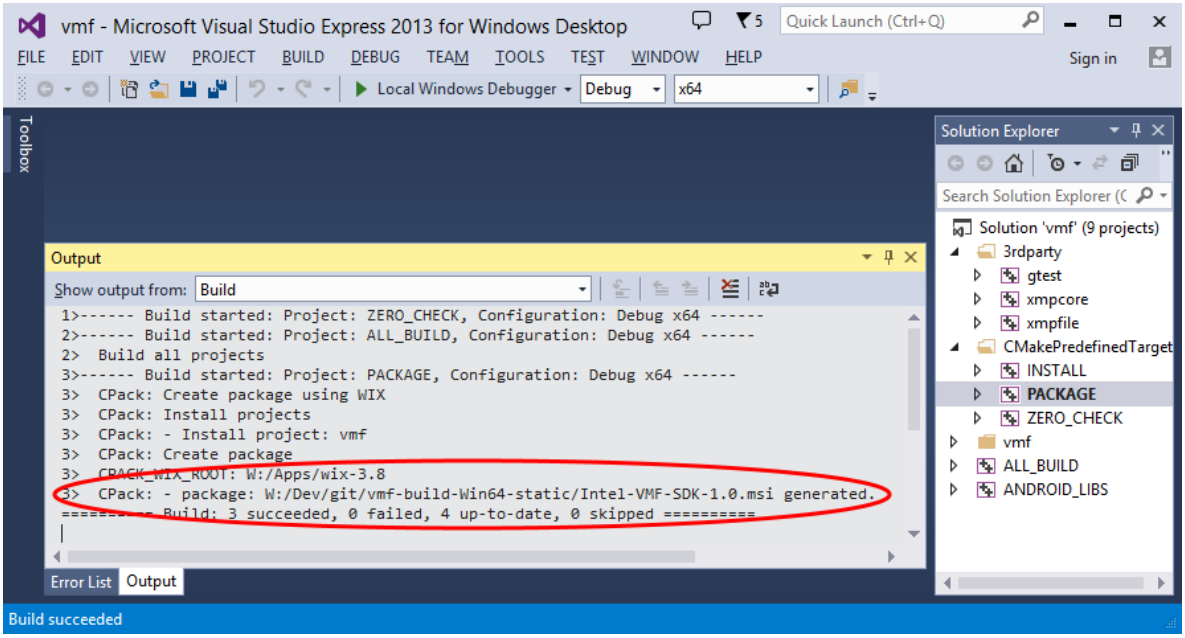
7. [Optional] You may add android libraries by building the ANDROID_LIBS project. It builds release and debug libraries for armeabi, armeabi-v7a and x86. In order to build this project right-click on it in Visual Studio and select the “Build” item. This can be done in one configuration only (ie “Debug” or “Release” – no need to rebuild in both configurations)



8. Change license by placing desired text into <SOURCES>/LICENSE.txt

9. Close Visual Studio, open CMake again and press “Configure” and then “Generate” buttons

10. Open vmf.sln from the folder set in the “Where to build the binaries” field in Visual Studio, select the “Release” configuration and build the PACKAGE target. Name of result package can be found in build log



If you want to build installer for several architectures (x86 and x64) and several MSVS versions (e.g., vc12 and vc14):

1. Choose which configuration will be built the last – it will create the final MSI package.
2. At first configure, generate solution and build INSTALL target for all the configurations you plan to include to the final package.
3. Create a "to_be_installed" folder in the top-level VMF **sources** folder.
4. Copy <Build Dir>\install\x64 and <Build Dir>\install\x86 folders to <VMF Source Dir>\to_be_installed.
5. Build PACKAGE target in the configuration you dedicated to be the last (see step 1).
6. Remove the <SOURCES>\to_be_installed folder.