# Installation Qt Creator and the components required to buildVisual Studio

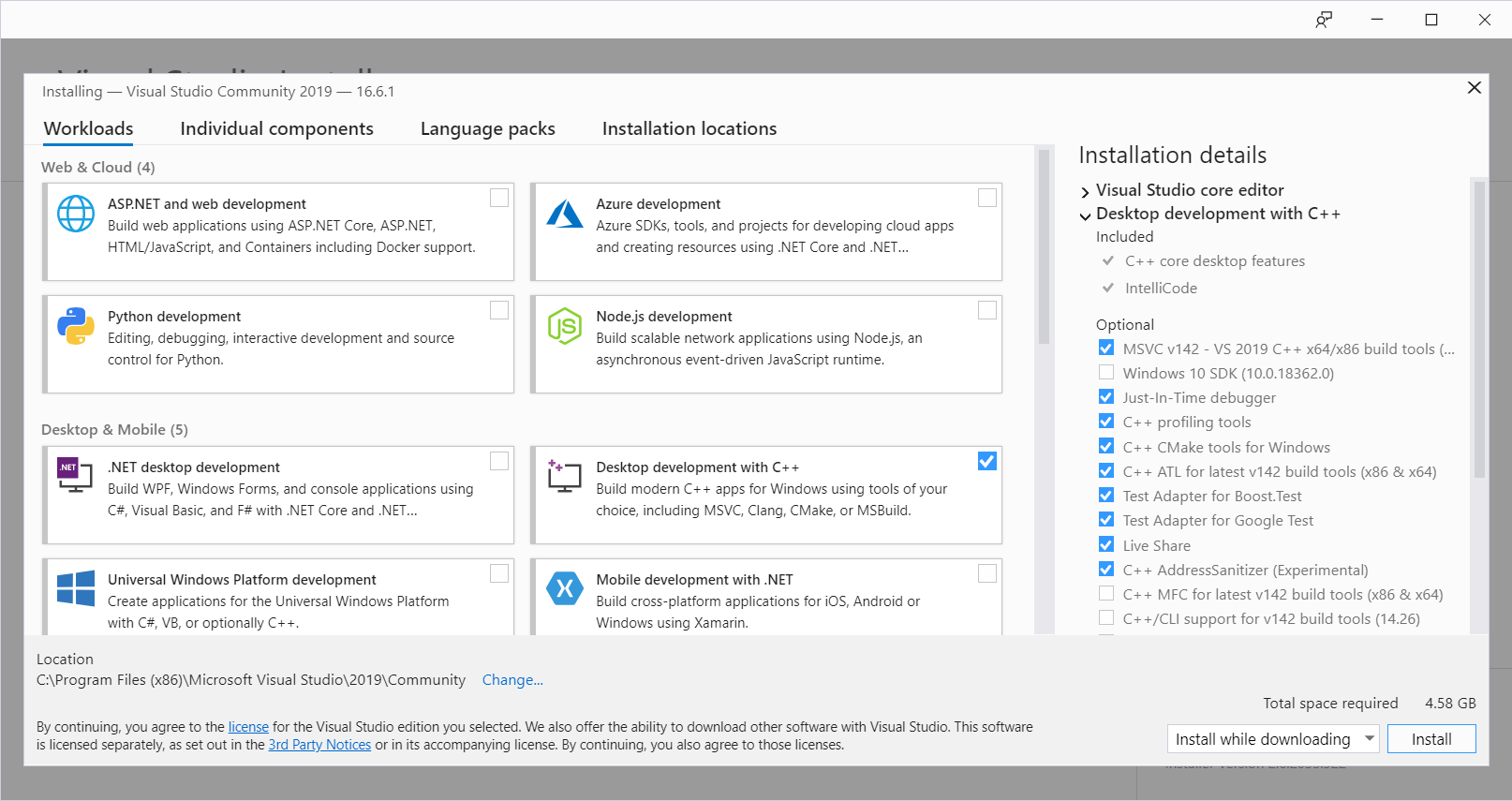
## Windows version

Installed under: Microsoft Windows [Version 10.0.19041.208], now updated to: Microsoft Windows [Version 10.0.19041.630]

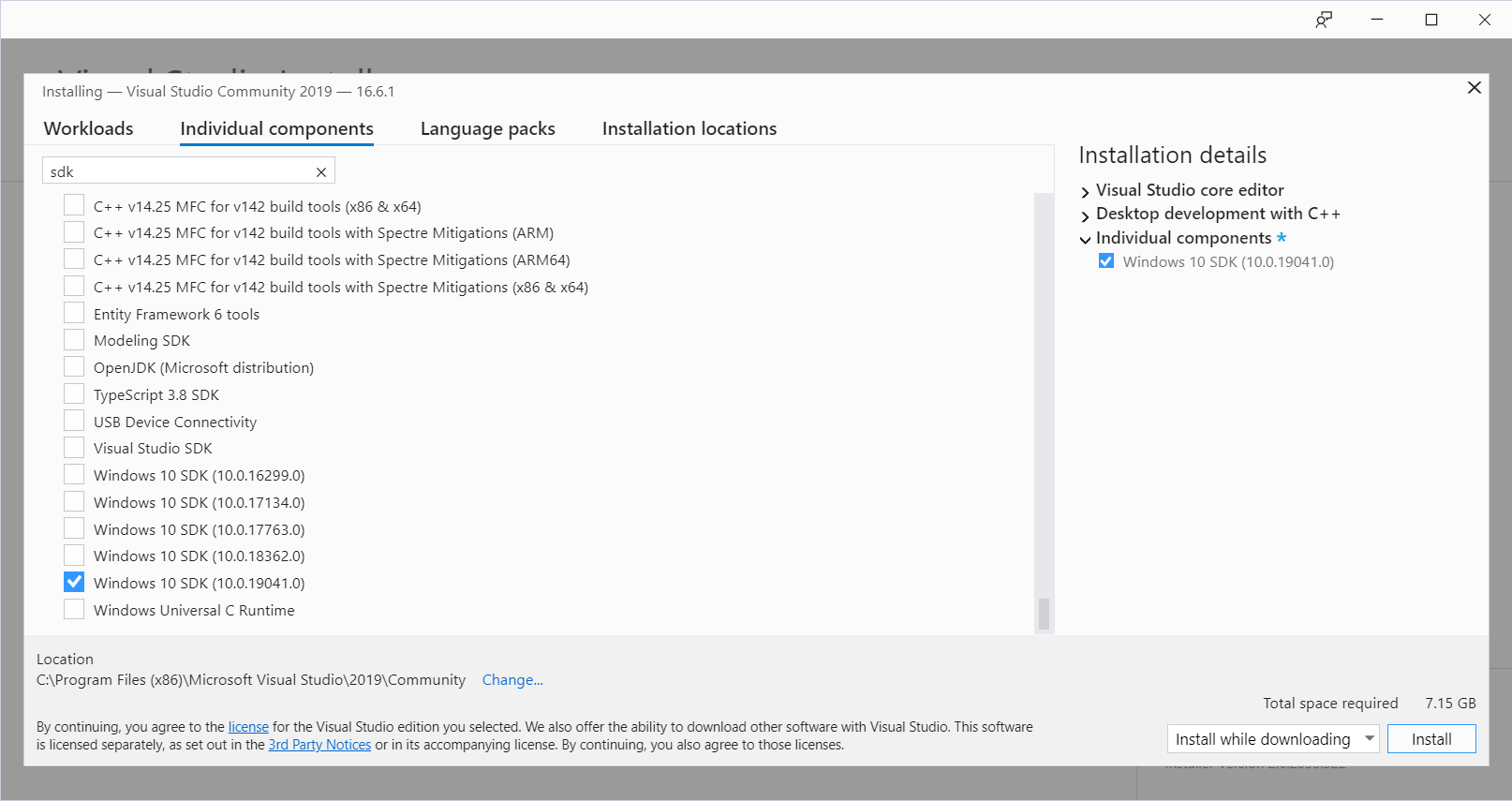
## Installing Visual Studio Community 2019

Installed: Visual Studio Community 2019 16.6.1

When installing, I chose in Workloads: **Desktop development with C ++**



Selected: Windows SDK 10.0.19041.0



## Installing Qt 5.9.5

Downloaded Qt 5.9.5 with:

<https://download.qt.io/official_releases/qt/5.9/5.9.5/>

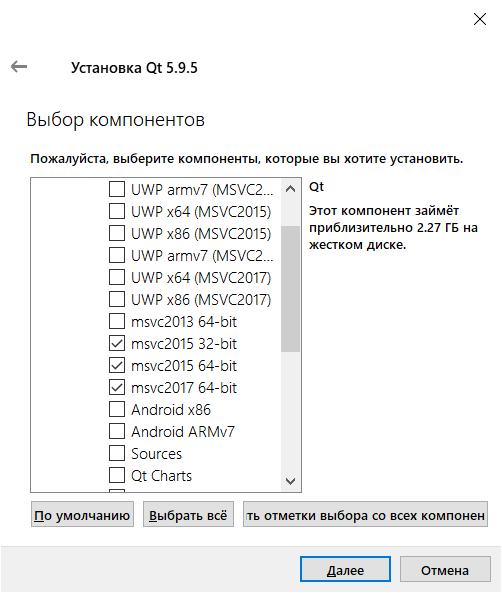
Namely 2 files:

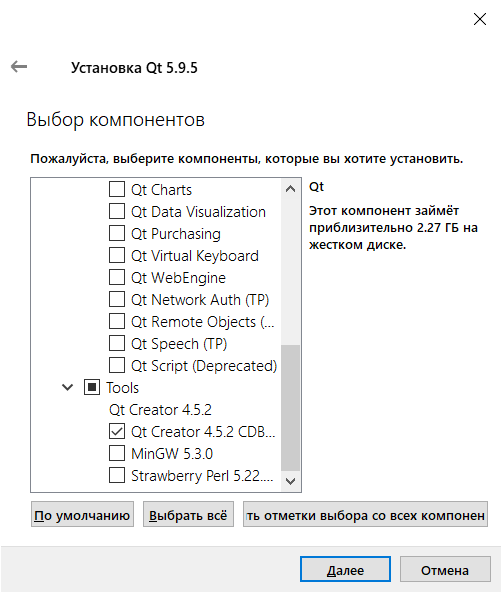
<https://download.qt.io/official_releases/qt/5.9/5.9.5/qt-opensource-windows-x86-5.9.5.exe>

<https://download.qt.io/official_releases/qt/5.9/5.9.5/qt-opensource-windows-x86-pdb-files-desktop-5.9.5.zip>

Installed Qt 5.9.5 with the following options

* msvc2015 32-bit
* msvc2015 64-bit
* msvc2017 64-bit



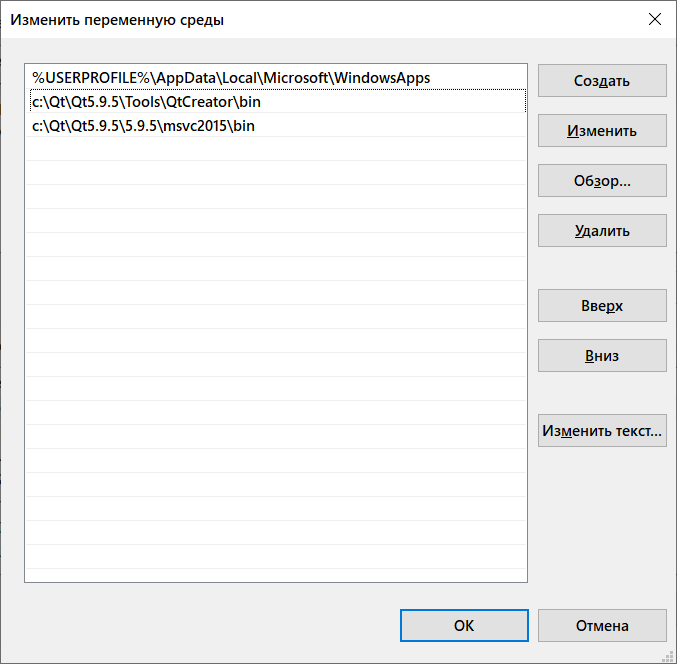


## Setting the PATH environment variable

Added directories to the user's PATH environment variable:

c:\Qt\Qt5.9.5\Tools\QtCreator\bin

c:\Qt\Qt5.9.5\5.9.5\msvc2015\bin



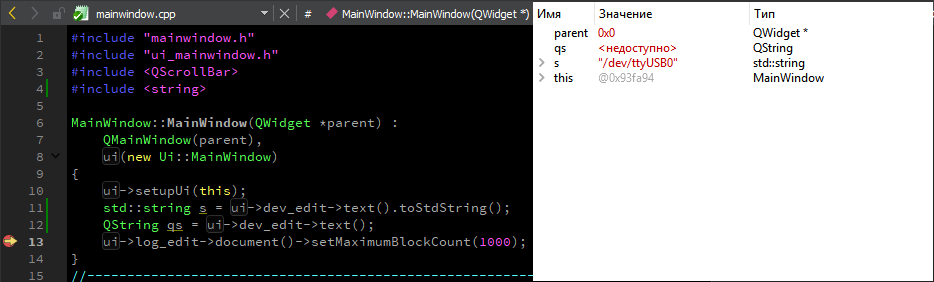
## Adding PDB Files

Currently, the PDB files required for debugging are provided in a separate zip file.

This zip file contains the 7zip files, which in turn must be extracted to the correct subfolder of the Qt installation.

Without PDB files, for example, the values ​​of composite variables are not displayed in a debugger such as QString.

In the window: "Locals and Expressions" in the values ​​of these variables is displayed: <not available> or <not accessible>



When installing the latest version of Qt through the online installer, you can choose: debug info.

For more details, see QTBUG-67587 with the resolution - Won't Do:

**PDB files of windows online installer are provided in a unmanageable huge compressed folder**

<https://bugreports.qt.io/browse/QTBUG-67587>

Well, and read how other developers relate to this:

**some Qt types in debugger shown as 'not accessible'**

<https://bugreports.qt.io/browse/QTCREATORBUG-19797>

[Permalink](https://bugreports.qt.io/browse/QTCREATORBUG-19797?focusedCommentId=427681&page=com.atlassian.jira.plugin.system.issuetabpanels:comment-tabpanel" \l "comment-427681" \o "Right click and copy link for a permanent link to this comment.)

[David Schulz](https://bugreports.qt.io/secure/ViewProfile.jspa?name=davschul)added a comment - 19 Oct '18 09:05

Unfortunately the pdb files aren't part of the offline packages anymore. Feel free to leave a vote onif this causes trouble for you.

[Permalink](https://bugreports.qt.io/browse/QTCREATORBUG-19797?focusedCommentId=429087&page=com.atlassian.jira.plugin.system.issuetabpanels:comment-tabpanel" \l "comment-429087" \o "Right click and copy link for a permanent link to this comment.)

[Permalink](https://bugreports.qt.io/browse/QTCREATORBUG-19797?focusedCommentId=434219&page=com.atlassian.jira.plugin.system.issuetabpanels:comment-tabpanel" \l "comment-434219" \o "Right click and copy link for a permanent link to this comment.)

[David Schulz](https://bugreports.qt.io/secure/ViewProfile.jspa?name=davschul)added a comment - 22 Nov '18 06:44

Feel free to leave a vote onif this causes trouble for you. I also think this is more than uncomfortable.

Because we installed Qt from the offline installer, you need to manually copy these files.

Unzipped the previously downloaded zip file with pdb files:

<http://download.qt.io/official_releases/qt/5.9/5.9.5/qt-opensource-windows-x86-pdb-files-desktop-5.9.5.zip>

For MSVC2015 X86\_64, do the following:

Selected .7z files by mask:

\* Windows-Windows\_10-MSVC2015-Windows-Windows\_10-X86\_64 \*. \*

Unzipped into one directory using Total Commander.

Allocated files.

Performed the command: Files -> Unpack ... (Alt + F9)

Copied to:

c:\Qt\Qt5.9.5\5.9.5\msvc2015\_64

Likewise, for the 32-bit version, copy the corresponding PDB files to: c:\Qt\Qt5.9.5\5.9.5\msvc2015msvc2015

## WebKit

In Qt 5.6.0 WebKit has been removed.

[https://github.com/goldendict/goldendict#building-under-windows-with-ms-visual-studio](https://github.com/goldendict/goldendict" \l "building-under-windows-with-ms-visual-studio)

**Building under Windows with MS Visual Studio**

Note: In Qt 5.6.0 and later theWebkitmodule was removed from official release builds. You should to build it from sources to compile GoldenDict.

I didn't compile from source.  
On: <https://github.com/qtwebkit/qtwebkit/releases>, there are compiled qtwebkit for versions only:

Qt 5.14.1, Qt 5.13, Qt 5.9

I downloaded the compiled QtWebKit for Qt 5.9:

QtWebKit 5.212.0 Alpha 2

<https://github.com/qtwebkit/qtwebkit/releases/tag/qtwebkit-5.212.0-alpha2>

<https://github.com/qtwebkit/qtwebkit/releases/download/qtwebkit-5.212.0-alpha2/qtwebkit-5.212.0_alpha2-qt59-msvc2015-x86.zip>

<https://github.com/qtwebkit/qtwebkit/releases/download/qtwebkit-5.212.0-alpha2/qtwebkit-5.212.0_alpha2-qt59-msvc2015-x64.zip>

MinGW is only 32-bit:

<https://github.com/qtwebkit/qtwebkit/releases/download/qtwebkit-5.212.0-alpha2/qtwebkit-5.212.0_alpha2-qt59-mingw530-x86.zip>

I expanded the archive into a folder:

c: \ Qt \ Qt5.9.5 \ 5.9.5 \ msvc2015

c: \ Qt \ Qt5.9.5 \ 5.9.5 \ msvc2015\_64

c: \ Qt \ Qt5.9.5 \ 5.9.5 \ mingw53\_32

## Launching Qt Creator

Before starting Qt Creator, you need to set environment variables for VS 2015.

Run command line via shortcut:

VS2015 x64 Native Tools Command Prompt

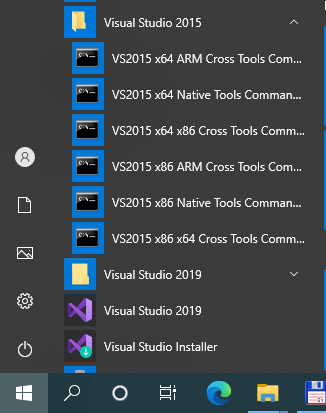
or

VS2015 x86 Native Tools Command Prompt, respectively for 32bit,

which respectively execute the commands:

% comspec% / k "" C: \ Program Files (x86) \ Microsoft Visual Studio 14.0 \ VC \ vcvarsall.bat "" amd64

% comspec% / k "" C: \ Program Files (x86) \ Microsoft Visual Studio 14.0 \ VC \ vcvarsall.bat "" x86



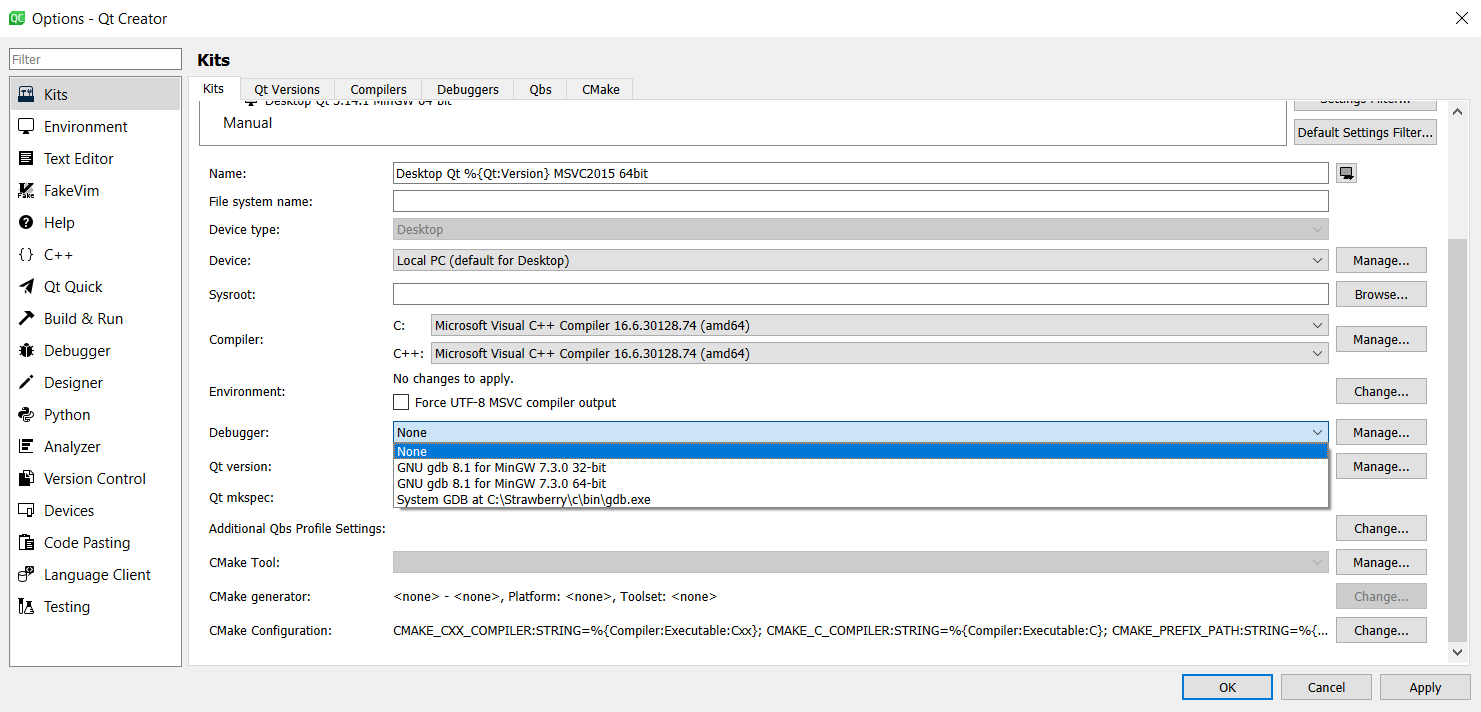
Then, in the command line that opens, run Qt Creator:

C:\Qt\Qt5.9.5\Tools\QtCreator\bin\qtcreator.exe

## Debugger cdb.exe is not available

It is necessary set fields for Compiler and Debugger.

Launched Qt Creator, but Debugger cdb.exe was not available.



## Windows 10 SDK

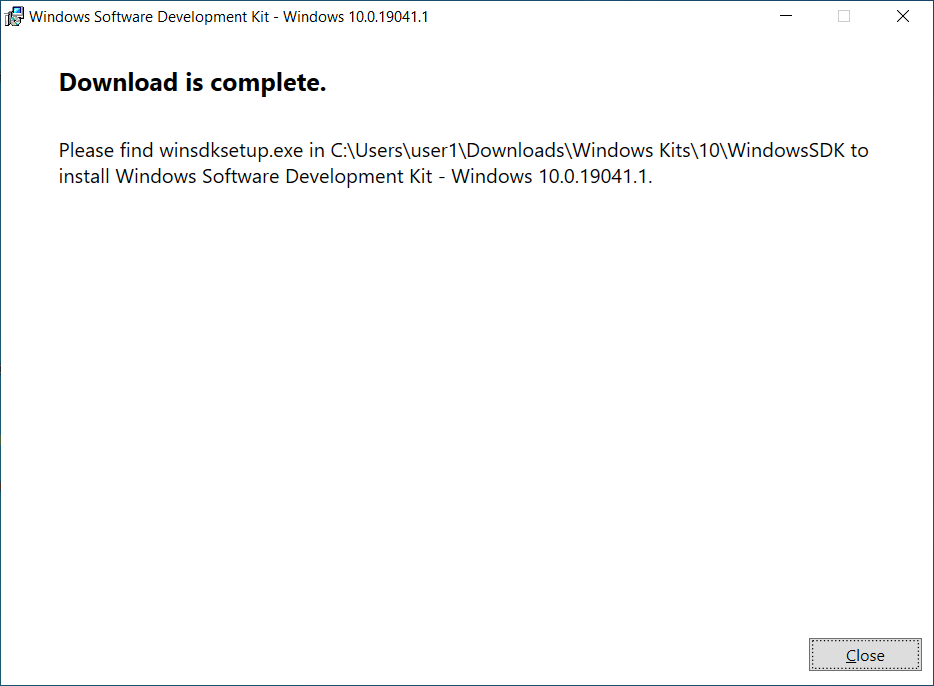
Downloaded Windows 10 SDK (10.0.19041.0)

<https://developer.microsoft.com/ru-ru/windows/downloads/windows-10-sdk/>

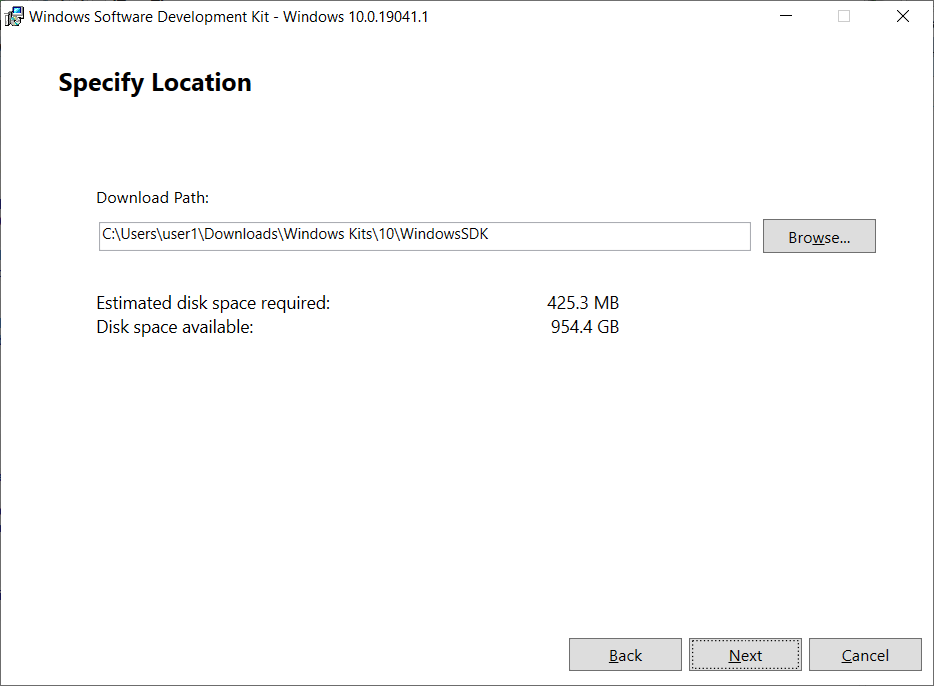
Installed only debuggers, from 2 msi files:

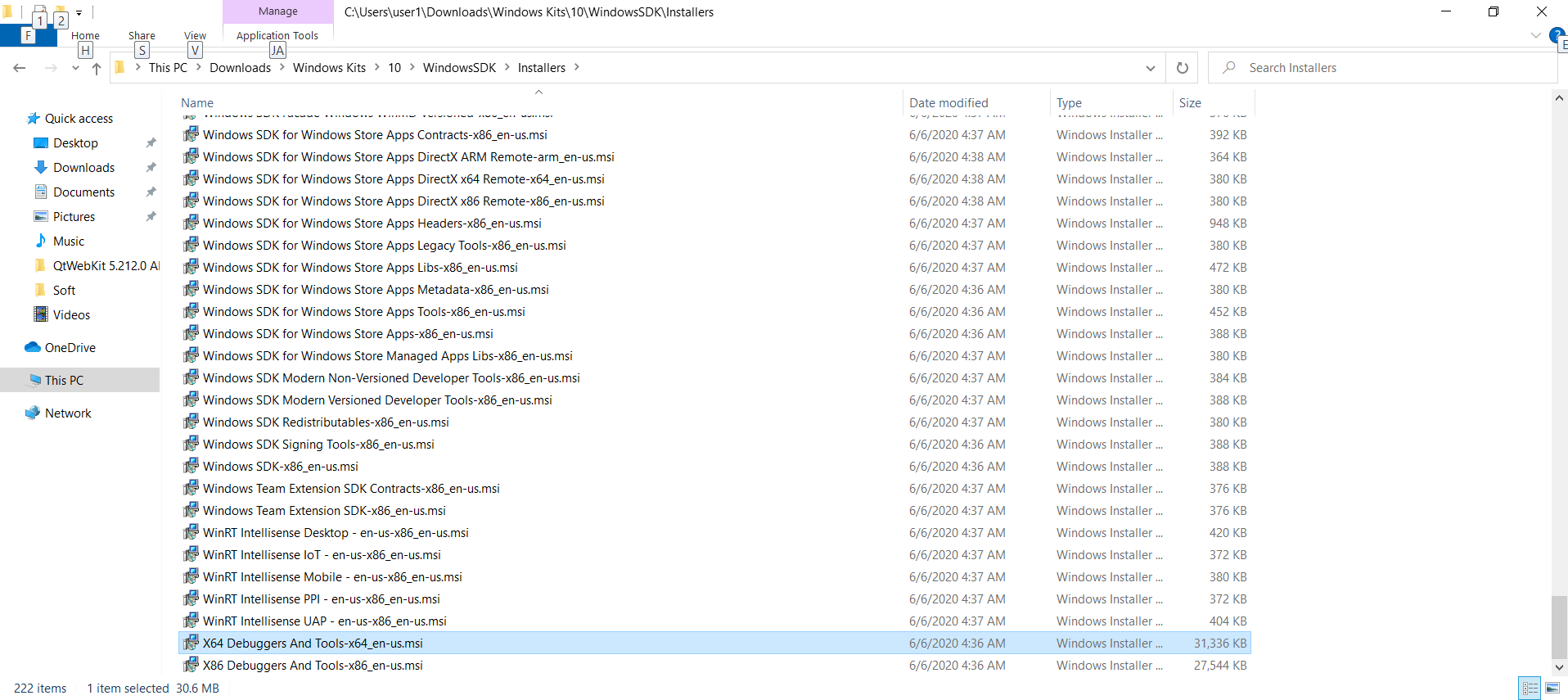
X64 Debuggers And Tools-x64\_en-us.msi

X86 Debuggers And Tools-x86\_en-us.msi



Directory: C:\Users\user1\Downloads\Windows Kits\10\WindowsSDK

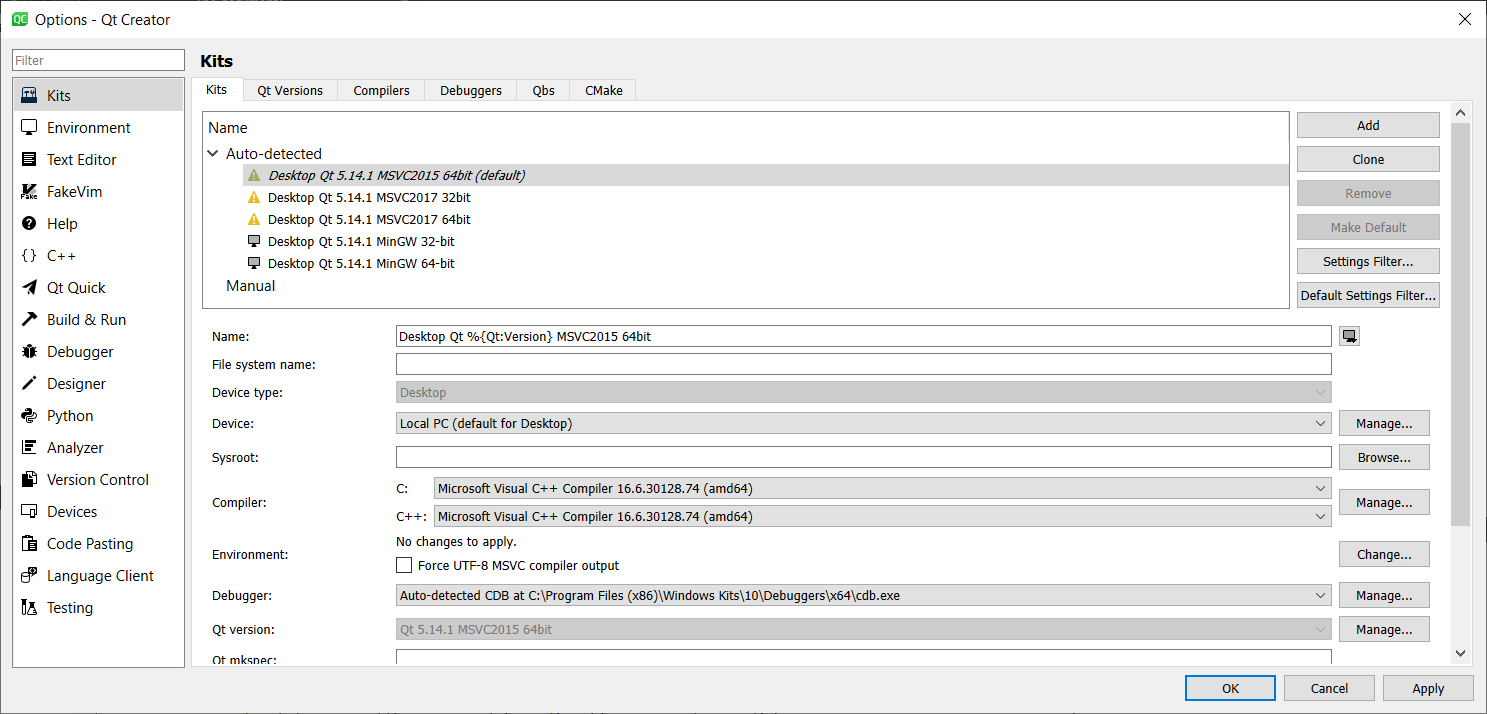




Installed:

X64 Debuggers And Tools-x64\_en-us.msi

The debugger appeared:



More details:

[https://doc.qt.io/qtcreator/creator-debugger-engines.html#debugging-tools-for-windows](https://doc.qt.io/qtcreator/creator-debugger-engines.html" \l "debugging-tools-for-windows)

### Debugging Tools for Windows

To use the CDB debugger, you must install the Debugging tools for Windows. You can download them from

[Download and Install Debugging Tools for Windows](https://developer.microsoft.com/windows/downloads/windows-10-sdk)as part of the Windows SDK.

**Note:**Visual Studio does not include the Debugging tools needed, and therefore, you must install them separately.

In addition, you must select Qt Creator CDB Debugger Support (in Qt> Tools> Qt Creator) when you install Qt or the stand-alone Qt Creator.

### AddingSymbol Server provided by Microsoft at symbol search path for Windows debuggers

Optional item.

To get debugging information for operating system libraries, for debugging Windows applications,

add the symbol server provided by Microsoft to the debugger symbol search path:

As a result of reading

<https://docs.microsoft.com/en-us/windows-hardware/drivers/debugger/symbol-path>

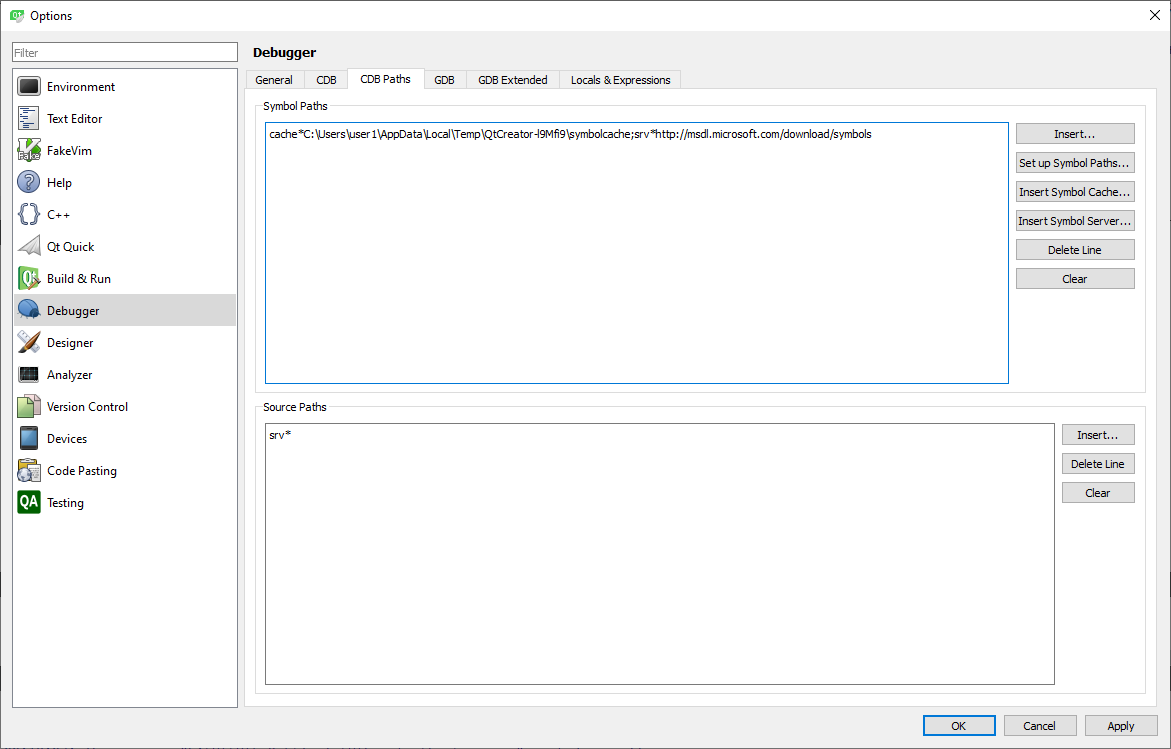
asked:

**cache \* C:\Users\user1\AppData\Local\Temp\QtCreator-l9Mfi9\symbolcache;srv\*http: //msdl.microsoft.com/download/symbols**

As in the example:

dbgcmdCopy

.sympath cache\*c:\MySymbols;srv\*https://msdl.microsoft.com/download/symbols



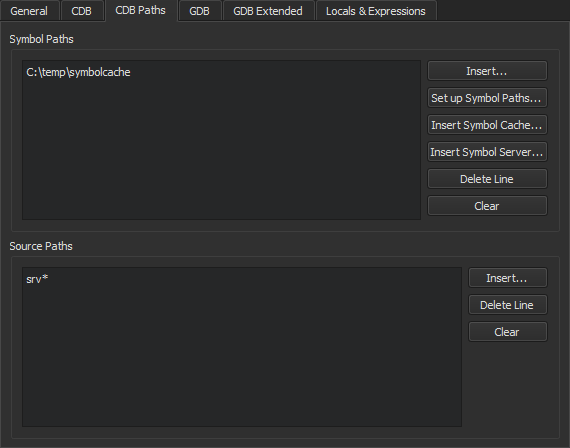
More details:

[https://doc.qt.io/qtcreator/creator-debugger-engines.html#setting-cdb-paths-on-windows](https://doc.qt.io/qtcreator/creator-debugger-engines.html" \l "setting-cdb-paths-on-windows)

## Setting CDB Paths on Windows

To obtain debugging information for the operating system libraries for debugging Windows applications, add the Symbol Server provided by Microsoft to the symbol search path of the debugger:

1. Select Tools> Options> Debugger> CDB Paths.



1. In the Symbol Paths group, select Insert.
2. Select the directory where you want to store the cached information.

Use a subfolder in a temporary directory, such asC: \ temp \ symbolcache...

1. Select OK.

**Note:**Populating the cache might take a long time on a slow network connection.

To use the Source Server infrastructure for fetching missing source files directly from version control or the web, enter the following string in the Source Paths field:srv \*...

## Git

Installed Git

<https://git-scm.com/download/win>