Setting Up Debugger

The main debugger settings are associated with the kit you build and run your project with. To specify the debugger and compiler to use for each kit, select **Edit** > **Preferences** > **Kits**.

You need to set up the debugger only if the automatic setup fails because the native debugger is missing (as is usually the case for the CDB debugger on Windows, which you always must install yourself) or because the installed version is not supported (for example, when your system contains no, or an outdated version of GDB and you want to use a locally installed replacement instead).

> **Note:** If you need to change the debugger to use for an automatically detected kit, you can **Clone** the kit and change the parameters in the clone. Make sure to select the cloned kit for your project.

If the debugger you want to use is not automatically detected, select **Edit** > **Preferences** > **Kits** > **Debuggers** > **Add** to add it.

> **Note:** To use the debugging tools for Windows, you must install them and add the Symbol Server provided by Microsoft to the symbol search path of the debugger. For more information, see Setting CDB Paths on Windows.

> **Note:** To use the Free Software Foundation (FSF) version of GDB on macOS, you must sign it and modify your kit settings.

This section explains the options you have for debugging C++ code and provides installation notes for the supported native debuggers. It also applies for code in other compiled languages such as C, FORTRAN, Ada.

For more information on the debugger modes, see Launching the Debugger in Different Modes.

Supported Native Debugger Versions

Qt Creator supports native debuggers when working with compiled code. On most supported platforms, the GNU Symbolic Debugger GDB can be used. On Microsoft Windows, when using the Microsoft tool chain, the Microsoft Console Debugger CDB is needed. On macOS and Linux, the LLDB debugger can be used.

The following table summarizes the support for debugging C++ code:

Platform	Compiler	Native Debugger
Linux	GCC, ICC	GDB, LLDB
Platform	**Compiler**	**Native Debugger**

Windows/MinGW	GCC	GDB
Windows/MSVC	Microsoft Visual C++ Compiler	Debugging Tools for Windows/CDB

Supported GDB Versions

Starting with version 3.1, Qt Creator requires the Python scripting extension. GDB builds without Python scripting are not supported anymore and will not work. The minimum supported version is GDB 7.5 using Python version 2.7, or 3.3, or newer.

For remote debugging using GDB and GDB server, the minimum supported version of GDB server on the target device is 7.0.

Supported CDB Versions

All versions of CDB targeting platforms supported by Qt are supported by Qt Creator.

Supported LLDB Versions

The LLDB native debugger has similar functionality to the GDB debugger. LLDB is the default debugger in Xcode on macOS for supporting C++ on the desktop. LLDB is typically used with the Clang compiler (even though you can use it with GCC, too).

On macOS you can use the LLDB version delivered with Xcode or build from source. The minimum supported version is LLDB 320.4.

On Linux, the minimum supported version is LLDB 3.8.

Installing Native Debuggers

The following sections provide information about installing native debuggers.

GDB

On Windows, use the Python-enabled GDB version that is bundled with the Qt package or comes with recent versions of MinGW. On most Linux distributions, the GDB builds shipped with the system are sufficient.

You can also build your own GDB, as instructed in Building GDB.

Builds of GDB shipped with Xcode on macOS are no longer supported.

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 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.

It is highly recommended that you add the Symbol Server provided by Microsoft to the symbol search path of the debugger. The Symbol Server provides you with debugging informaton for the operating system libraries for debugging Windows applications. For more information, see Setting CDB Paths on Windows.

Debugging Tools for macOS

The Qt binary distribution contains both debug and release variants of the libraries. But you have to explicitly tell the runtime linker that you want to use the debug libraries even if your application is compiled as debug, as release is the default library.

If you use a qmake based project in Qt Creator, you can set a flag in your run configuration, in **Projects** mode. In the run configuration, select **Use debug version of frameworks**.

For more detailed information about debugging on macOS, see: Mac OS X Debugging Magic.

LLDB

We recommend using the LLDB version that is delivered with the latest Xcode.

Setting up FSF GDB for macOS

To use FSF GDB on macOS, you must sign it and add it to the Qt Creator kits.

1. To create a key for signing FSF GDB, select **Keychain Access** > **Certificate Assistant** > **Create a Certificate**:
    1. In the **Name** field, input **fsfgdb** to replace the existing content.
    2. In the **Certificate Type** field, select **Code Signing**.
    3. Select the **Let me override defaults** check box.
    4. Select **Continue**, and follow the instructions of the wizard (use the default settings), until the **Specify a Location For The Certificate** dialog opens.
    5. In the **Keychain** field, select **System**.
    6. Select **Keychain Access** > **System**, and locate the certificate.
    7. Double click the certificate to view certificate information.
    8. In the **Trust** section, select **Always Trust** in the **When using this certificate** field, and then close the dialog.
2. To sign the binary, enter the following command in the terminal:

```
codesign -f -s "fsfgdb" $INSTALL_LOCATION/fsfgdb
```

3. In Qt Creator, select **Qt Creator** > **Preferences** > **Kits** > **Add** to create a kit that uses FSF GDB.
4. In the **Debugger** field, specify the path to FSF GDB (`$HOME/gdb72/bin/fsfgdb`, but with an explicit value for `$HOME`).
5. To use the debugger, add the kit in the **Build Settings** of the project.

‹ Debugging Launching the Debugger ›

Qt DOCUMENTATION