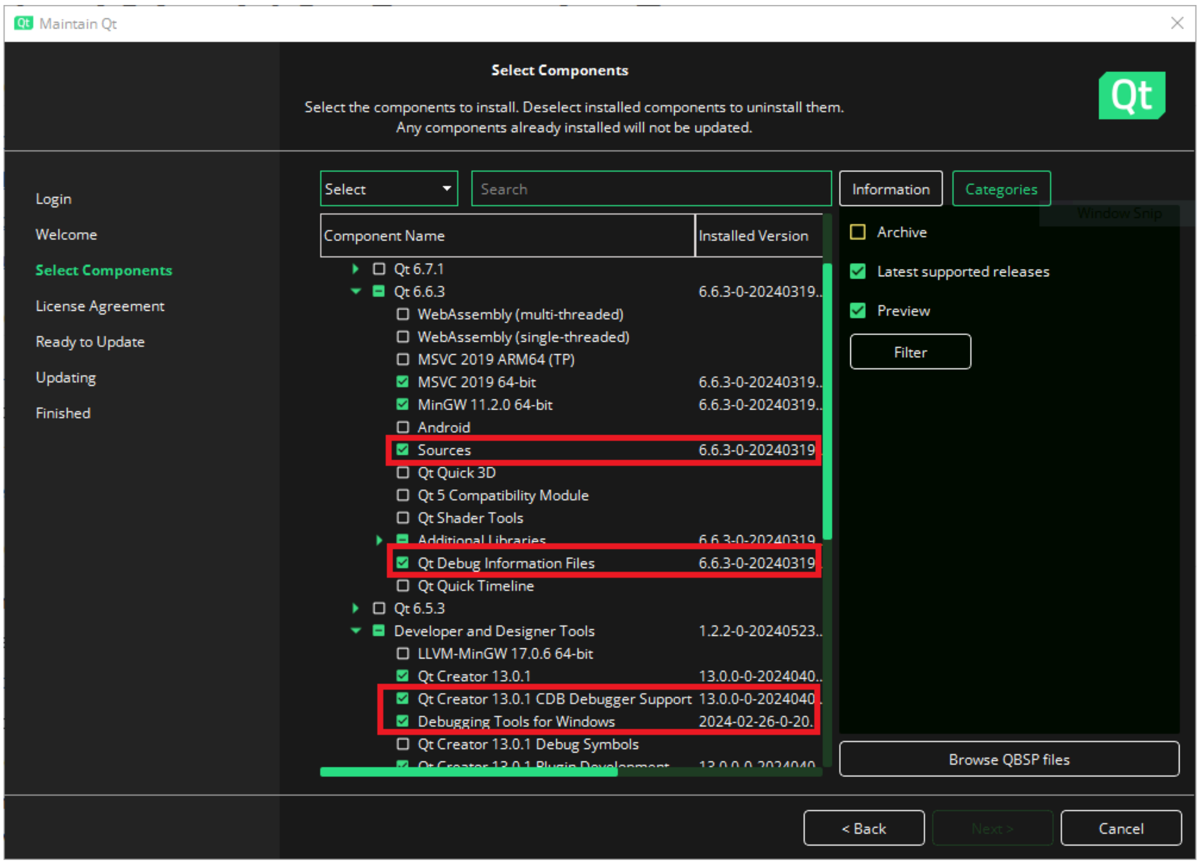
Debugging in Qt Creator

Qt requirements

Maintenance wizard

1. Sources (usually optional)
2. Qt Debug information (usually optional)
3. CDB Debugger Support (usually selected by default)
4. Debugging Tools for Windows (usually selected by default)



Select the components to install. Deselect installed components to uninstall them.
Any components already installed will not be updated.

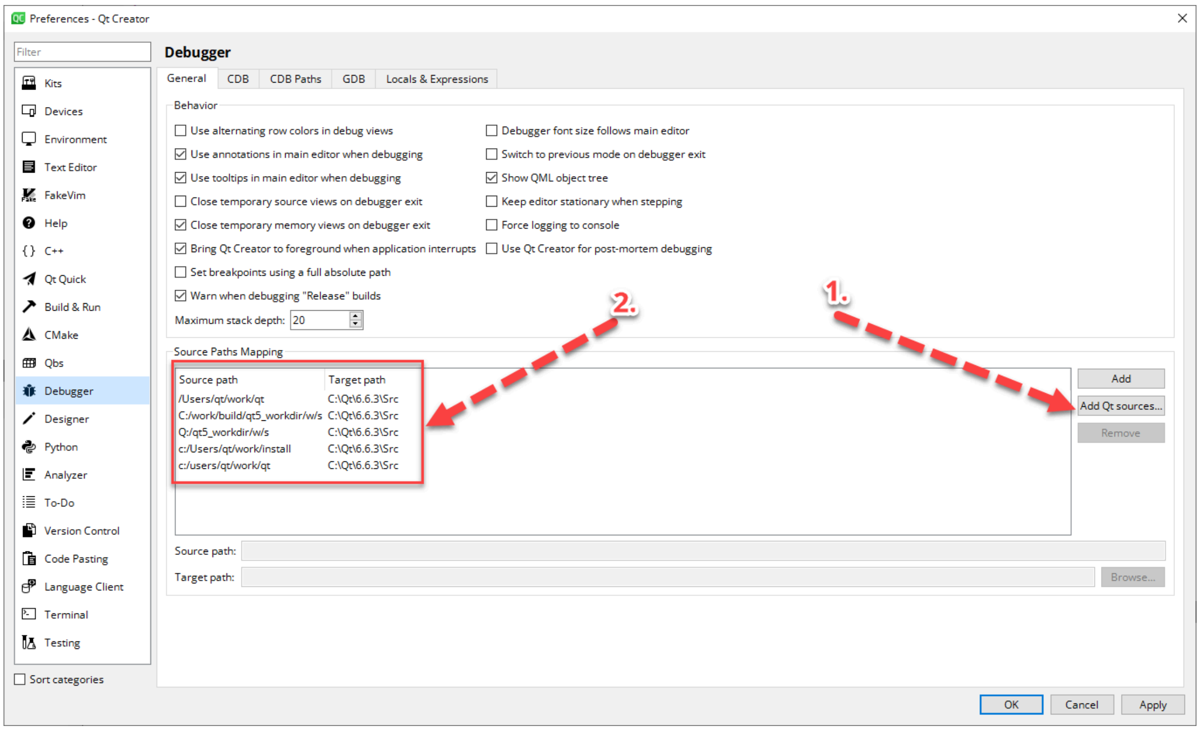
Component Name	Installed Version
☐ Qt 6.7.1	
▨ Qt 6.6.3	6.6.3-0-20240319..
☐ WebAssembly (multi-threaded)	
☐ WebAssembly (single-threaded)	
☐ MSVC 2019 ARM64 (TP)	
✓ MSVC 2019 64-bit	6.6.3-0-20240319..
✓ MinGW 11.2.0 64-bit	6.6.3-0-20240319..
☐ Android	
✓ Sources	6.6.3-0-20240319
☐ Qt Quick 3D	
☐ Qt 5 Compatibility Module	
☐ Qt Shader Tools	
▨ Additional Libraries	6.6.3-0-20240319
✓ Qt Debug Information Files	6.6.3-0-20240319
☐ Qt Quick Timeline	
☐ Qt 6.5.3	
▨ Developer and Designer Tools	1.2.2-0-20240523..
☐ LLVM-MinGW 17.0.6 64-bit	
✓ Qt Creator 13.0.1	13.0.0-0-2024040..
✓ Qt Creator 13.0.1 CDB Debugger Support	13.0.0-0-2024040
✓ Debugging Tools for Windows	2024-02-26-0-20.
☐ Qt Creator 13.0.1 Debug Symbols	
✓ Qt Creator 13.0.1 Plugin Development	13.0.0-0-2024040

☐ Archive
✓ Latest supported releases
✓ Preview

Qt Creator configuration

Configure sources

Edit -> Preferences -> Debugger -> Add Qt sources -> Select C:\Qt<VERSION>\Src



Debugger

Behavior

- [ ] Use alternating row colors in debug views
- [x] Use annotations in main editor when debugging
- [x] Use tooltips in main editor when debugging
- [ ] Close temporary source views on debugger exit
- [x] Close temporary memory views on debugger exit
- [x] Bring Qt Creator to foreground when application interrupts
- [ ] Set breakpoints using a full absolute path
- [x] Warn when debugging "Release" builds
- [ ] Debugger font size follows main editor
- [ ] Switch to previous mode on debugger exit
- [x] Show QML object tree
- [ ] Keep editor stationary when stepping
- [ ] Force logging to console
- [ ] Use Qt Creator for post-mortem debugging

Maximum stack depth: 20

Source Paths Mapping

Source path	Target path
/Users/qt/work/qt	C:\Qt\6.6.3\Src
C:/work/build/qt5_workdir/w/s	C:\Qt\6.6.3\Src
Q:/qt5_workdir/w/s	C:\Qt\6.6.3\Src
c:/Users/qt/work/install	C:\Qt\6.6.3\Src
c:/users/qt/work/qt	C:\Qt\6.6.3\Src

Source path:
Target path:

Memory analysis

Windows symbols environment variables and symbol path

Name: _NT_SYMBOL_PATH
Variable: SRV*C:\SymbolCache*https://msdl.microsoft.com/download/symbols

Setup for umdh

```
gflags /i [IMAGE.EXE] +ust
gflags /i [IMAGE.EXE] +ust
```

Check the key in regedit: Computer\HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Image File Execution Options

To change the database size call gflags from the commandline:

- Select 'Image tab'
- Enter image name, including the .exe suffix
- Hit 'Tab' key to populate
- Select 'Stack Backtrace (Megs)' at the bottom left and enter your value

Run umdh:

```
tlist | findstr *myapp.exe*
umdh -p:[PID] -f:before.txt
umdh -p:[PID] -f:after.txt
umdh before.txt after.txt > delta.txt
```

Use Windows calculator in programmer mode to conver to decimal

Qt test harness slots:

```
void MemoryManagement::initTestCase()
{
    quint64 pid = QCoreApplication::applicationPid();
    QString umdh32 = "C:\\Program Files (x86)\\Windows
Kits\\10\\Debuggers\\x86\\umdh.exe";

    QStringList argList;
    QString pidArg = "-p:" % QString::number(pid);
    QString fileArg = "-f:before.txt";
    argList << pidArg << fileArg;

    QProcess process;
    process.setProgram(umdh32);
    process.setArguments(argList);
    process.start();
    process.waitForFinished();

}

void MemoryManagement::cleanupTestCase()
{
    quint64 pid = QCoreApplication::applicationPid();
    QString umdh32 = "C:\\Program Files (x86)\\Windows
Kits\\10\\Debuggers\\x86\\umdh.exe";

    QStringList argList;
    QString pidArg = "-p:" % QString::number(pid);
    QString fileArg = "-f:after.txt";
    argList << pidArg << fileArg;

    QProcess process1;
    process1.setProgram(umdh32);
    process1.setArguments(argList);
    process1.start();
    process1.waitForFinished();

    QProcess process2;
    QString compareDumps = "C:\\Program Files (x86)\\Windows
Kits\\10\\Debuggers\\x86\\umdh.exe";
    argList.clear();
    argList << "before.txt" << "after.txt";
    process2.setProgram(compareDumps);
    process2.setArguments(argList);
    process2.start();
    process2.waitForFinished();

    QByteArray deltaString = process2.readAllStandardOutput();

    QFile f( "delta.txt" );
    if( !f.open(QIODevice::WriteOnly) ){
        qInfo() << "Couldn't open delta.txt for writing";
        return;
    }
    QTextStream ts( &f );
    ts << deltaString;

    f.close();
    }
```