# Edit and Continue (Visual C++)

05/31/2017 3 minutes to read Contributors all

## In this article

You can use Edit and Continue in Visual C++ projects. See Supported Code Changes (C++) for information about the limitations of Edit and Continue.

For more information about Visual Studio 2015 Update 3 improvements, see C++ Edit and Continue in Visual Studio 2015 Update 3.

The /Zo (Enhance Optimized Debugging) compiler option that was introduced in Visual Studio 2013 Update 3 adds additional information to .pdb (symbol) files for binaries compiled without the /Od (Disable (Debug)) option.

**/Zo** disables Edit and Continue. See How to: Debug Optimized Code.

## Enable or disable Edit and Continue

You may want to disable the automatic invocation of Edit and Continue if you are making edits to the code that you do not want applied during the current debugging session. You can also re-enable automatic Edit and Continue.

**Important**

For required build settings and other information about feature compatibility, see [C++ Edit and Continue in Visual Studio 2015 Update 3](**https://blogs.msdn.microsoft.com/vcblog/2016/07/01/c-edit-and-continue-in-visual-studio-2015-update-3/**.

1. If you are in a debugging session, stop debugging (**Shift + F5**).

2. On the **Tools** menu, choose **Options**.

3. In the **Options** dialog box, select **Debugging > General**.

4. To enable, select **Enable Edit and Continue**. To disable, clear the checkbox.

5. In the **Edit and Continue** group, select or clear the **Enable Native Edit and Continue** check box.

   Altering this setting affects all projects you work on. You do not need to rebuild your application after changing this setting. If you build your application from the command line or from a makefile, but you debug in the Visual Studio environment, you can still use Edit and Continue if you set the **/ZI** option.

## How to apply code changes explicitly

In Visual C++, Edit and Continue can apply code changes in two ways. Code changes can be applied implicitly, when you choose an execution command, or explicitly, using the **Apply Code Changes** command.

When you apply code changes explicitly, your program remains in break mode - no execution occurs.

- To apply code changes explicitly, on the **Debug** menu, choose **Apply Code Changes**.

## How to stop code changes

While Edit and Continue is in the process of applying code changes, you can stop the operation.

To stop applying code changes:

- On the **Debug** menu, choose **Stop Applying Code Changes**.

  This menu item is visible only when code changes are being applied.

  If you choose this option, none of the code changes are committed.

## How to reset the point of execution

Some code changes can cause the point of execution to move to a new location when Edit and Continue applies the changes. Edit and Continue places the point of execution as accurately as possible, but the results may not be correct in all cases.

In Visual C++, a dialog box informs you when the point of execution changes. You should verify that the location is correct before you continue debugging. If it is not correct, use the **Set Next Statement** command. For more information, see Set the next statement to execute.

## How to work with stale code

In some cases, Edit and Continue cannot apply code changes to the executable immediately, but might be able to apply the code changes later if you continue debugging. This happens if you edit a function that calls the current function or if you add more than 64 bytes of new variables to a function on the call stack

In such cases, the debugger continues executing the original code until the changes can be applied. The stale code appears as a temporary source file window in a separate source window, with a title such as `enc25.tmp`. The edited source continues to appear in the original source window. If you try to edit the stale code, a warning message appears.

## See Also

Supported Code Changes (C++)