**Compile and build in Visual Studio**

Running a build creates assemblies and executable applications from your source code at any point during a development cycle. In general, the build process is very similar across many different project types such as Windows, ASP.NET, mobile apps, and others. The build process is also very similar across programming languages such as C#, Visual Basic, C++, and F#.

By building your code often, you can quickly identify compile-time errors, such as incorrect syntax, misspelled keywords, and type mismatches. You can also quickly detect and correct run-time errors, such as logic errors and semantic errors, by frequently building and running debug versions of the code.

A successful build is essentially a validation that the application's source code contains correct syntax and that all static references to libraries, assemblies, and other components have been resolved. This produces an application executable that can then be tested for proper functioning in both a [debugging environment](https://docs.microsoft.com/en-us/visualstudio/debugger/index?view=vs-2017) and through a variety of manual and automated tests to [validate code quality](https://docs.microsoft.com/en-us/visualstudio/test/improve-code-quality?view=vs-2017). Once the application has been fully tested, you can then compile a release version to deploy to your customers. For an introduction to this process, see [Walkthrough: Building an application](https://docs.microsoft.com/en-us/visualstudio/ide/walkthrough-building-an-application?view=vs-2017).

You can use any of the following methods to build an application: the Visual Studio IDE, the MSBuild command-line tools, and Azure Pipelines:

| **Build Method** | **Benefits** |
| --- | --- |
| IDE | - Create builds immediately and test them in a debugger. - Run multi-processor builds for C++ and C# projects. - Customize different aspects of the build system. |
| MSBuild command line | - Build projects without installing Visual Studio. - Run multi-processor builds for all project types. - Customize most areas of the build system. |
| Azure Pipelines | - Automate your build process as part of a continuous integration/continuous delivery pipeline. - Apply automated tests with every build. - Employ virtually unlimited cloud-based resources for build processes. - Modify the build workflow and create build activities to perform deeply customized tasks. |

The documentation in this section goes into further details of the IDE-based build process. For more information on the other methods, see [MSBuild](https://docs.microsoft.com/en-us/visualstudio/msbuild/msbuild?view=vs-2017) and [Azure Pipelines](https://docs.microsoft.com/en-us/azure/devops/pipelines/index?view=vsts), respectively.

## Overview of building from the IDE

When you create a project, Visual Studio created default build configurations for the project and the solution that contains the project. These configurations define how the solutions and projects are built and deployed. Project configurations in particular are unique for a target platform (such as Windows or Linux) and build type (such as debug or release). You can edit these configurations however you like, and can also create your own configurations as needed.

For a first introduction to building within the IDE, see [Walkthrough: Building an application](https://docs.microsoft.com/en-us/visualstudio/ide/walkthrough-building-an-application?view=vs-2017).

Next, see [Building and cleaning projects and solutions in Visual Studio](https://docs.microsoft.com/en-us/visualstudio/ide/building-and-cleaning-projects-and-solutions-in-visual-studio?view=vs-2017) to learn about the different aspects customizations you can make to the process. Customizations include [changing output directories](https://docs.microsoft.com/en-us/visualstudio/ide/how-to-change-the-build-output-directory?view=vs-2017), [specifying custom build events](https://docs.microsoft.com/en-us/visualstudio/ide/specifying-custom-build-events-in-visual-studio?view=vs-2017), [managing project dependencies](https://docs.microsoft.com/en-us/visualstudio/ide/how-to-create-and-remove-project-dependencies?view=vs-2017), [managing build log files](https://docs.microsoft.com/en-us/visualstudio/ide/how-to-view-save-and-configure-build-log-files?view=vs-2017), and [suppressing compiler warnings](https://docs.microsoft.com/en-us/visualstudio/ide/how-to-suppress-compiler-warnings?view=vs-2017).

From there, you can explore a variety of other tasks:

* [Understand build configurations](https://docs.microsoft.com/en-us/visualstudio/ide/understanding-build-configurations?view=vs-2017)
* [Understand build platforms](https://docs.microsoft.com/en-us/visualstudio/ide/understanding-build-platforms?view=vs-2017)
* [Manage project and solution properties](https://docs.microsoft.com/en-us/visualstudio/ide/managing-project-and-solution-properties?view=vs-2017).
* Specify build events in [C#](https://docs.microsoft.com/en-us/visualstudio/ide/how-to-specify-build-events-csharp?view=vs-2017) and [Visual Basic](https://docs.microsoft.com/en-us/visualstudio/ide/how-to-specify-build-events-visual-basic?view=vs-2017).
* [Set build options](https://docs.microsoft.com/en-us/visualstudio/ide/reference/options-dialog-box-projects-and-solutions-build-and-run?view=vs-2017)
* [Build multiple projects in parallel](https://docs.microsoft.com/en-us/visualstudio/msbuild/building-multiple-projects-in-parallel-with-msbuild?view=vs-2017).

**Walkthrough: Build an application**

By completing this walkthrough, you'll become more familiar with several options that you can configure when you build applications with Visual Studio. You'll create a custom build configuration, hide certain warning messages, and increase build output information for a sample application.

**Install the sample application**

Download the [Introduction to building WPF applications](https://code.msdn.microsoft.com/Introduction-to-Building-b8d16419) sample. Choose either C# or Visual Basic. After the *.zip* file has downloaded, extract it and open the *ExpenseItIntro.sln* file using Visual Studio.

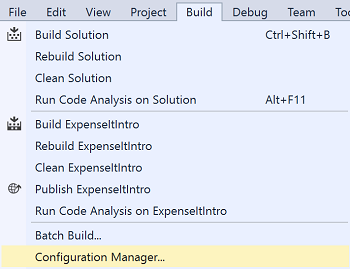
**Create a custom build configuration**

When you create a solution, debug and release build configurations and their default platform targets are defined for the solution automatically. You can then customize these configurations or create your own. Build configurations specify the build type. Build platforms specify the operating system that an application targets for that configuration. For more information, see [Understand build configurations](https://docs.microsoft.com/en-us/visualstudio/ide/understanding-build-configurations?view=vs-2017), [Understand build platforms](https://docs.microsoft.com/en-us/visualstudio/ide/understanding-build-platforms?view=vs-2017), and [How to: Set debug and release configurations](https://docs.microsoft.com/en-us/visualstudio/debugger/how-to-set-debug-and-release-configurations?view=vs-2017).

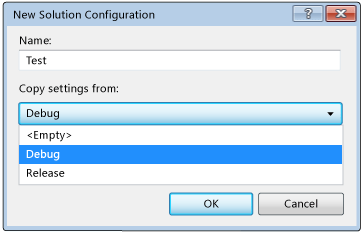
You can change or create configurations and platform settings by using the **Configuration Manager** dialog box. In this procedure, you'll create a build configuration for testing.

**Create a build configuration**

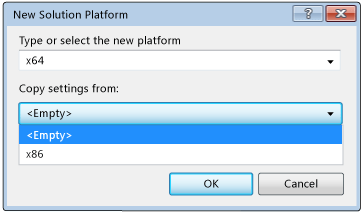
1. Open the **Configuration Manager** dialog box.



1. In the **Active solution configuration** list, choose **<New...>**.
2. In the **New Solution Configuration** dialog box, name the new configuration Test, copy settings from the existing **Debug** configuration, and then choose the **OK** button.

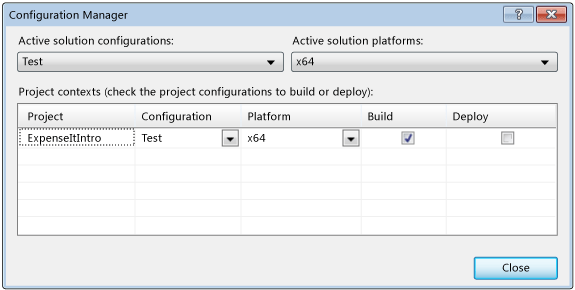


1. In the **Active solution platform** list, choose **<New...>**.
2. In the **New Solution Platform** dialog box, choose **x64**, and don't copy settings from the x86 platform.



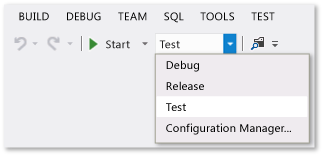
1. Choose the **OK** button.

The active solution configuration has been changed to **Test** with the active solution platform set to x64.



1. Choose **Close**.

You can quickly verify or change the active solution configuration by using the **Solution Configurations** list on the **Standard** toolbar.



**Build the application**

Next, you'll build the solution with the custom build configuration.

**Build the solution**

* On the menu bar, choose **Build** > **Build Solution**.

The **Output** window displays the results of the build. The build succeeded.

**Hide compiler warnings**

Next we'll introduce some code that causes a warning to be generated by the compiler.

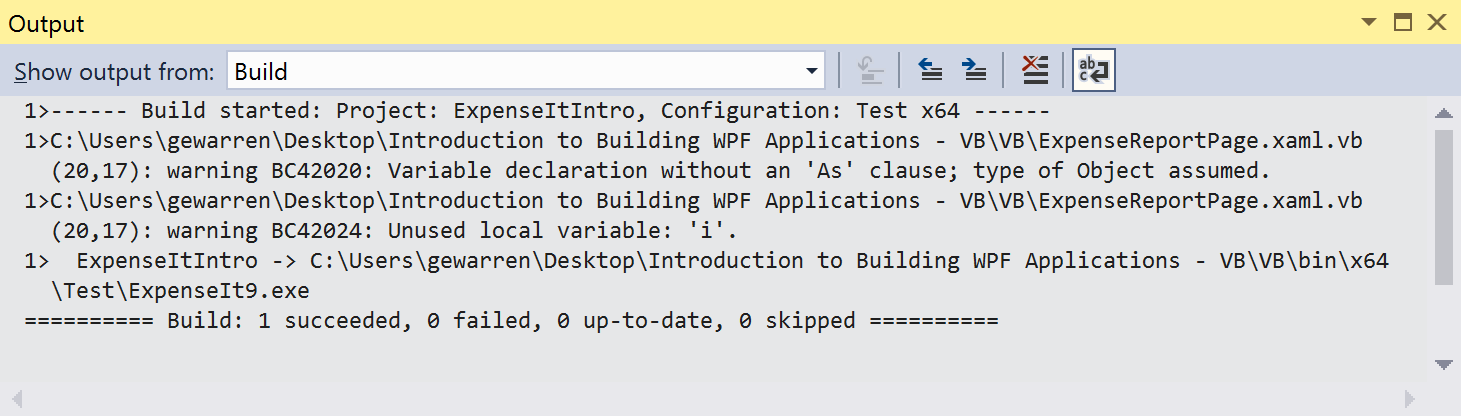
1. In the C# project, open the *ExpenseReportPage.xaml.cs* file. In the **ExpenseReportPage** method, add the following code: int i;.

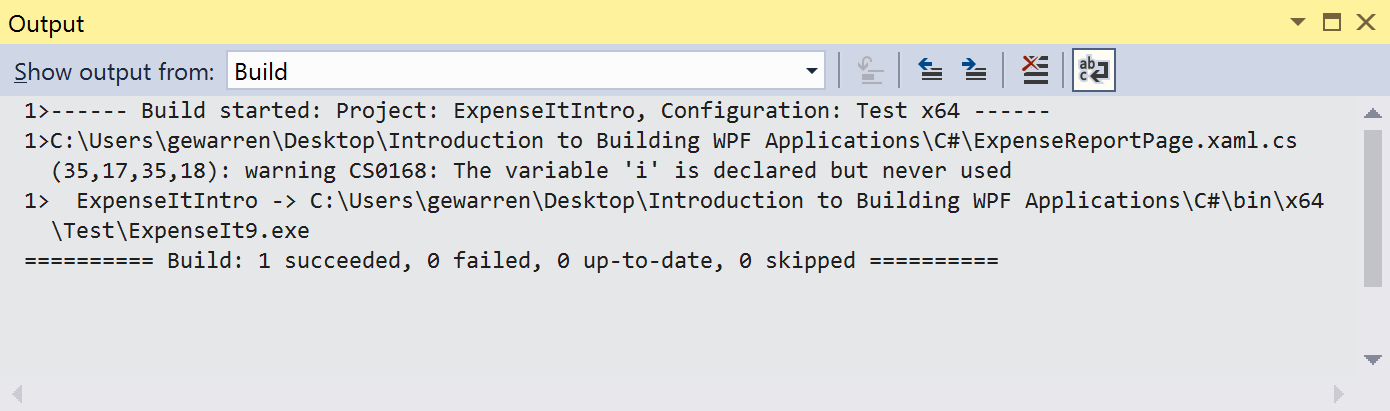
OR

In the Visual Basic project, open the *ExpenseReportPage.xaml.vb* file. In the custom constructor **Public Sub New...**, add the following code: Dim i.

1. Build the solution.

The **Output** window displays the results of the build. The build succeeded, but warnings were generated:





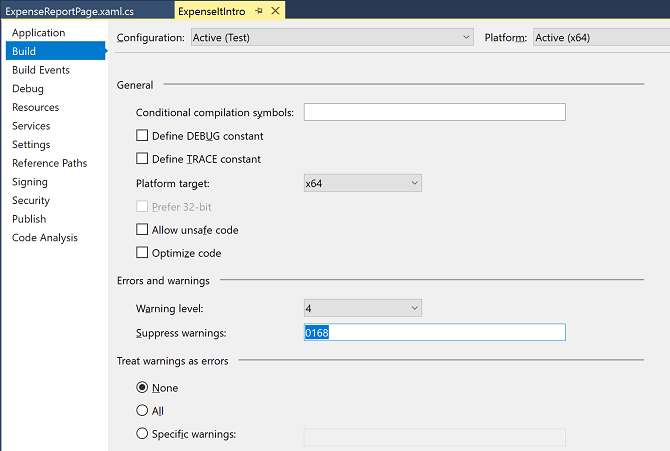
You can temporarily hide certain warning messages during a build rather than have them clutter up the build output.

**Hide a specific C# warning**

1. In **Solution Explorer**, choose the top-level project node.
2. On the menu bar, choose **View** > **Property Pages**.

The **Project Designer** opens.

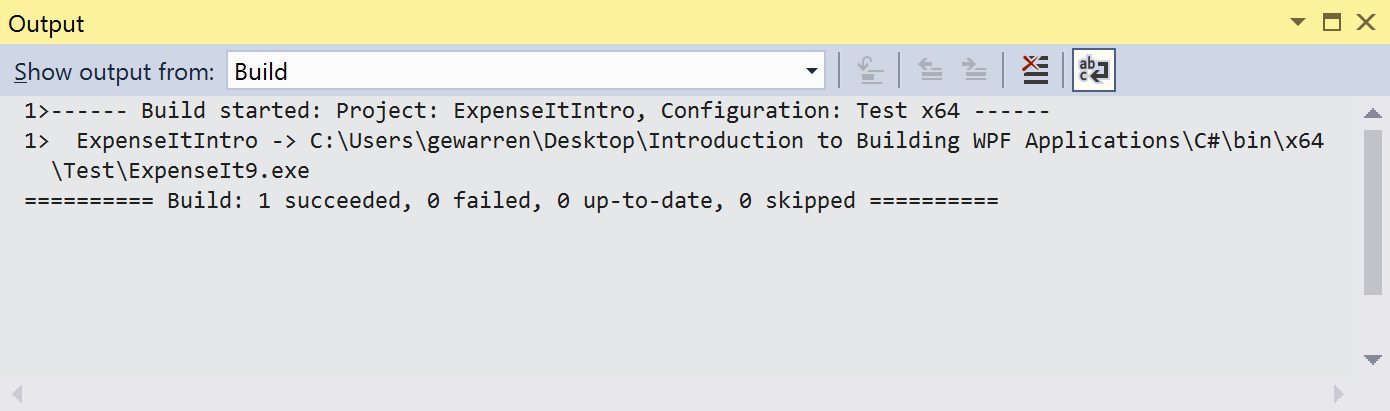
1. Choose the **Build** page and then, in the **Suppress warnings** box, specify the warning number **0168**.



For more information, see [Build Page, Project Designer (C#)](https://docs.microsoft.com/en-us/visualstudio/ide/reference/build-page-project-designer-csharp?view=vs-2017).

1. Build the solution.

The **Output** window displays only summary information for the build.

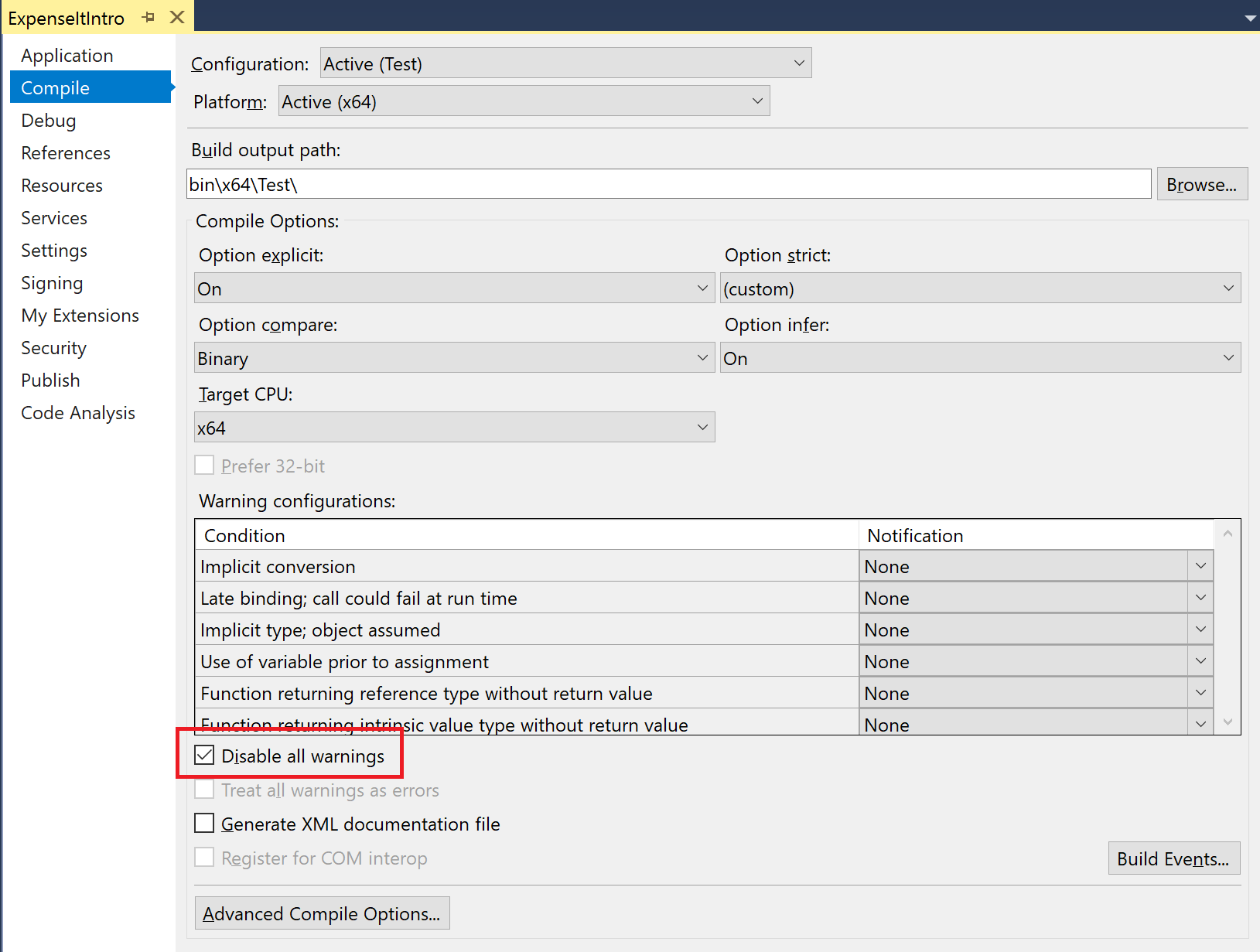


**Suppress all Visual Basic build warnings**

1. In **Solution Explorer**, choose the top-level project node.
2. On the menu bar, choose **View** > **Property Pages**.

The **Project Designer** opens.

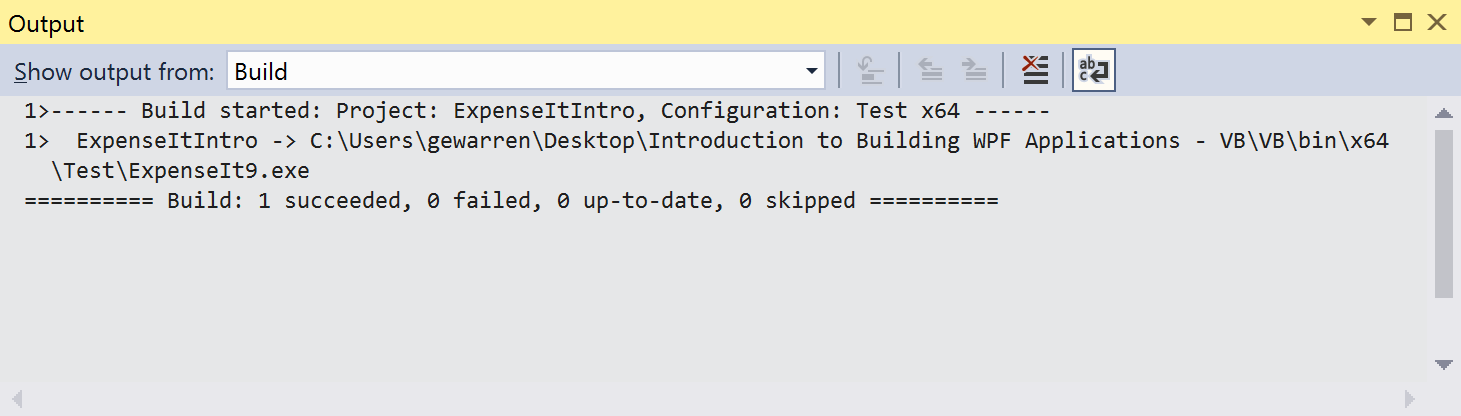
1. On the **Compile** page, select the **Disable all warnings** check box.



For more information, see [Configure warnings in Visual Basic](https://docs.microsoft.com/en-us/visualstudio/ide/configuring-warnings-in-visual-basic?view=vs-2017).

1. Build the solution.

The **Output** window displays only summary information for the build.



For more information, see [How to: Suppress compiler warnings](https://docs.microsoft.com/en-us/visualstudio/ide/how-to-suppress-compiler-warnings?view=vs-2017).

**Display additional build details in the Output window**

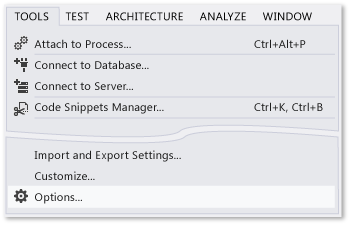
You can change how much information about the build process appears in the **Output** window. Build verbosity is usually set to **Minimal**, which means that the **Output** window displays only a summary of the build process along with any high priority warnings or errors. You can display more information about the build by using the [Options dialog box, Projects and Solutions, Build and Run](https://docs.microsoft.com/en-us/visualstudio/ide/reference/options-dialog-box-projects-and-solutions-build-and-run?view=vs-2017).

Important

If you display more information, the build will take longer to complete.

**Change the amount of information in the Output window**

1. Open the **Options** dialog box.



1. Choose the **Projects and Solutions** category, and then choose the **Build and Run** page.
2. In the **MSBuild project build output verbosity** list, choose **Normal**, and then choose the **OK** button.
3. On the menu bar, choose **Build** > **Clean Solution**.
4. Build the solution, and then review the information in the **Output** window.

The build information includes the time that the build started (located at the beginning) and the order in which files were processed. This information also includes the actual compiler syntax that Visual Studio runs during the build.

For example, in the C# build, the [/nowarn](https://docs.microsoft.com/en-us/dotnet/visual-basic/reference/command-line-compiler/nowarn) option lists the warning code, **1762**, that you specified earlier in this topic, along with three other warnings.

In the Visual Basic build, [/nowarn](https://docs.microsoft.com/en-us/dotnet/visual-basic/reference/command-line-compiler/nowarn) doesn't include specific warnings to exclude, so no warnings appear.

Tip

You can search the contents of the **Output** window if you display the **Find** dialog box by choosing the **Ctrl**+**F** keys.

For more information, see [How to: View, save, and configure build log files](https://docs.microsoft.com/en-us/visualstudio/ide/how-to-view-save-and-configure-build-log-files?view=vs-2017).

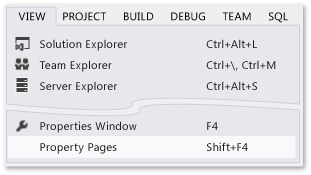
**Create a Release Build**

You can build a version of the sample application that's optimized for shipping it. For the release build, you'll specify that the executable is copied to a network share before the build is kicked off.

For more information, see [How to: Change the build output directory](https://docs.microsoft.com/en-us/visualstudio/ide/how-to-change-the-build-output-directory?view=vs-2017) and [Build and clean projects and solutions in Visual Studio](https://docs.microsoft.com/en-us/visualstudio/ide/building-and-cleaning-projects-and-solutions-in-visual-studio?view=vs-2017).

**Specify a release build for Visual Basic**

1. Open the **Project Designer**.



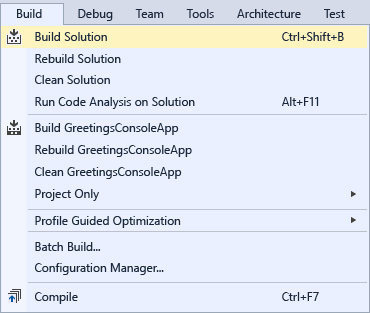
1. Choose the **Compile** page.
2. In the **Configuration** list, choose **Release**.
3. In the **Platform** list, choose **x86**.
4. In the **Build output path** box, specify a network path.

For example, you can specify \\myserver\builds.

Important

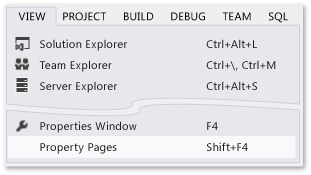
A message box might appear, warning you that the network share that you've specified might not be a trusted location. If you trust the location that you've specified, choose the **OK** button in the message box.

1. Build the application.



**Specify a release build for C#**

1. Open the **Project Designer**.



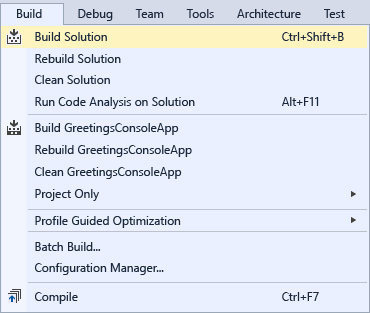
1. Choose the **Build** page.
2. In the **Configuration** list, choose **Release**.
3. In the **Platform** list, choose **x86**.
4. In the **Output path** box, specify a network path.

For example, you could specify \\myserver\builds.

Important

A message box might appear, warning you that the network share that you've specified might not be a trusted location. If you trust the location that you've specified, choose the **OK** button in the message box.

1. On the **Standard toolbar**, set the Solution Configurations to **Release** and the Solution Platforms to **x86**.
2. Build the application.



The executable file is copied to the network path that you specified. Its path would be \\myserver\builds\\FileName.exe.

**Building and cleaning projects and solutions in Visual Studio**

### In this article

1. [To build, rebuild, or clean an entire solution](https://docs.microsoft.com/en-us/visualstudio/ide/building-and-cleaning-projects-and-solutions-in-visual-studio?view=vs-2017#to-build-rebuild-or-clean-an-entire-solution)
2. [To build or rebuild a single project](https://docs.microsoft.com/en-us/visualstudio/ide/building-and-cleaning-projects-and-solutions-in-visual-studio?view=vs-2017#to-build-or-rebuild-a-single-project)
3. [To build only the startup project and its dependencies](https://docs.microsoft.com/en-us/visualstudio/ide/building-and-cleaning-projects-and-solutions-in-visual-studio?view=vs-2017#to-build-only-the-startup-project-and-its-dependencies)
4. [To build only the selected Visual C++ project](https://docs.microsoft.com/en-us/visualstudio/ide/building-and-cleaning-projects-and-solutions-in-visual-studio?view=vs-2017#to-build-only-the-selected-visual-c-project)
5. [To compile multiple C++ project items](https://docs.microsoft.com/en-us/visualstudio/ide/building-and-cleaning-projects-and-solutions-in-visual-studio?view=vs-2017#to-compile-multiple-c-project-items)
6. [To stop a build](https://docs.microsoft.com/en-us/visualstudio/ide/building-and-cleaning-projects-and-solutions-in-visual-studio?view=vs-2017#to-stop-a-build)
7. [See also](https://docs.microsoft.com/en-us/visualstudio/ide/building-and-cleaning-projects-and-solutions-in-visual-studio?view=vs-2017#see-also)

By using the procedures in this topic, you can build, rebuild, or clean all or some of the projects or project items in a solution. For a step-by-step tutorial, see [Walkthrough: Building an application](https://docs.microsoft.com/en-us/visualstudio/ide/walkthrough-building-an-application?view=vs-2017).

**Note**

The UI in your edition of Visual Studio might differ from what this topic describes, depending on your active settings. To change your settings, for example to **General** or **Visual C++** settings, choose **Tools** > **Import and Export Settings**, and then choose **Reset all settings**.

## To build, rebuild, or clean an entire solution

1. In **Solution Explorer**, choose or open the solution.
2. On the menu bar, choose **Build**, and then choose one of the following commands:
   * Choose **Build** or **Build Solution** to compile only those project files and components that have changed since the most recent build.

**Note**

The **Build** command becomes **Build Solution** when a solution includes more than one project.

* + Choose **Rebuild Solution** to "clean" the solution and then build all project files and components.
  + Choose **Clean Solution** to delete any intermediate and output files. With only the project and component files left, new instances of the intermediate and output files can then be built.

## To build or rebuild a single project

1. In **Solution Explorer**, choose or open the project.
2. On the menu bar, choose **Build**, and then choose either **Build** ProjectName or **Rebuild** ProjectName.
   * Choose **Build** ProjectName to build only those project components that have changed since the most recent build.
   * Choose **Rebuild** ProjectName to "clean" the project and then build the project files and all project components.

## To build only the startup project and its dependencies

1. On the menu bar, choose **Tools** > **Options**.
2. In the **Options** dialog box, expand the **Projects and Solutions** node, and then choose the **Build and Run** page.

The **Build and Run** > **Projects and Solutions** > **Options** dialog box opens.

1. Select the **Only build startup projects and dependencies on Run** check box.

When this check box is selected, only the current startup project and its dependencies are built when you perform either of the following steps:

* + On the menu bar, choose **Debug** > **Start** (**F5**).
  + On the menu bar, choose **Build** > **Build Solution** (**Ctrl**+**Shift**+**B**).

When this check box is cleared, all projects, their dependencies, and the solution files are built when you run either of the preceding commands. By default, this check box is cleared.

## To build only the selected Visual C++ project

Choose a Visual C++ project, and then, on the menu bar, choose **Build** > **Project Only**, and one of the following commands:

* **Build Only** ProjectName
* **Rebuild Only** ProjectName
* **Clean Only** ProjectName
* **Link Only** ProjectName

These commands apply only to the Visual C++ project that you chose, without building, rebuilding, cleaning, or linking any project dependencies or solution files. Depending on your version of Visual Studio, the **Project Only** submenu might contain more commands.

## To compile multiple C++ project items

In **Solution Explorer**, choose multiple files that have can be compiled actions, open the shortcut menu for one of those files, and then choose **Compile**.

If the files have dependencies, the files will be compiled in dependency order. The compile operation will fail if the files require a precompiled header that isn't available when you compile. The compile operation uses the current active solution configuration.

## To stop a build

Perform either of the following steps:

* On the menu bar, select **Build** > **Cancel**.
* Press **Ctrl**+**Break**.

**How to: Change the build output directory**

You can specify the location of output on a per-configuration basis (for debug, release, or both) generated by your project.

**Note**

If you have a **Setup** project, see the note at the end of this article.

## Change the build output directory

1. On the menu bar, choose **Project** > **<Appname> Properties**. You can also right-click the project node in **Solution Explorer** and select **Properties**.
2. If you have a Visual Basic project, select the **Compile** tab. If you have a C# project, select the **Build** tab. If you have a C++ project or a JavaScript project, select the **General** tab.
3. In the configuration drop-down at the top, choose the configuration whose output file location you want to change (debug, release, or all).

Find the output path entry (**Build output path** in Visual Basic, **Output directory** in Visual C++, **Output path** in JavaScript and C#). Specify a new build output directory relative to the project directory.

**Note**

In a Setup Project, the **Output file name** box changes only the location of the Setup.exe file, not the location of the project files. For more information, see **Build, Configuration Properties, Deployment Project Properties dialog box**.

**How to: Build to a common output directory**

By default, Visual Studio builds each project in a solution in its own folder inside the solution. You can change the build output paths of your projects to force all outputs to be placed in the same folder.

## To place all solution outputs in a common directory

1. Click on one project in the solution.
2. On the **Project** menu, click **Properties**.
3. Depending on the type of project, click on either the **Compile** tab or the **Build** tab, and set the **Output path** to a folder to use for all projects in the solution.
4. Repeat steps 1-3 for all projects in the solution.

**Specify custom build events in Visual Studio**

By specifying a custom build event, you can automatically run commands before a build starts or after it finishes. For example, you can run a .bat file before a build starts or copy new files to a folder after the build is complete. Build events run only if the build successfully reaches those points in the build process.

For specific information about the programming language that you're using, see the following topics:

* Visual Basic--[How to: Specify build events (Visual Basic)](https://docs.microsoft.com/en-us/visualstudio/ide/how-to-specify-build-events-visual-basic?view=vs-2017).
* C# and F#--[How to: Specify build events (C#)](https://docs.microsoft.com/en-us/visualstudio/ide/how-to-specify-build-events-csharp?view=vs-2017).
* Visual C++--[Specify build events](https://docs.microsoft.com/cpp/ide/specifying-build-events).

## Syntax

Build events follow the same syntax as DOS commands, but you can use macros to create build events more easily. For a list of available macros, see [Pre-build Event/Post-build Event command line dialog box](https://docs.microsoft.com/en-us/visualstudio/ide/reference/pre-build-event-post-build-event-command-line-dialog-box?view=vs-2017).

For best results, follow these formatting tips:

* Add a call statement before all build events that run .bat files.

Example: call C:\MyFile.bat

Example: call C:\MyFile.bat call C:\MyFile2.bat

* Enclose file paths in quotation marks.

Example (for Windows 8): "%ProgramFiles(x86)%\Microsoft SDKs\Windows\v8.0A\Bin\NETFX 4.0 Tools\gacutil.exe" -if "$(TargetPath)"

* Separate multiple commands by using line breaks.
* Include wildcards as needed.

Example: for %I in (\*.txt \*.doc \*.html) do copy %I c:\mydirectory\

**Note**

%I in the code above should be %%I in batch scripts.

**How to: Set multiple startup projects**

Visual Studio allows you to specify how more than one project is run when you start the debugger.

## To set multiple startup projects

1. In **Solution Explorer**, select the solution (the top node).
2. Choose the solution node's context (right-click) menu and then choose **Properties**. The **Solution Property Pages** dialog box appears.
3. Expand the **Common Properties** node, and choose **Startup Project**.
4. Choose the **Multiple Startup Projects** option and set the appropriate actions.

**How to: Create and remove project dependencies**

When building a solution that contains multiple projects, it can be necessary to build certain projects first, to generate code used by other projects. When a project consumes executable code generated by another project, the project that generates the code is referred to as a project dependency of the project that consumes the code. Such dependency relationships can be defined in the **Project Dependencies** dialog box.

## To assign dependencies to projects

1. In **Solution Explorer**, select a project.
2. On the **Project** menu, choose **Project Dependencies**.

The **Project Dependencies** dialog box opens.

**Note**

The **Project Dependencies** option is only available in a solution with more than one project.

1. On the **Dependencies** tab, select a project from the **Project** drop-down menu.
2. In the **Depends on** field, select the check box of any other project that must build before this project does.

Your solution must consist of more than one project before you can create project dependencies.

## To remove dependencies from projects

1. In **Solution Explorer**, select a project.
2. On the **Project** menu, choose **Project Dependencies**.

The **Project Dependencies** dialog box opens.

**Note**

The **Project Dependencies** option is only available in a solution with more than one project.

1. On the **Dependencies** tab, select a project from the **Project** drop-down menu.
2. In the **Depends on** field, clear the check boxes beside any other projects that are no longer dependencies of this project.

**How to: View, save, and configure build log files**

After you build a project in the Visual Studio IDE, you can view information about that build in the **Output** window. By using this information, you can, for example, troubleshoot a build failure. For C++ projects, you can also view the same information in a .txt file that's created and saved automatically. For managed-code projects, you can copy and paste the information from the **Output** window into a .txt file and save it yourself. You can also use the IDE to specify what kinds of information you want to view about each build.

If you build any kind of project by using MSBuild, you can create a .txt file to save information about the build. For more information, see [Obtain build logs](https://docs.microsoft.com/en-us/visualstudio/msbuild/obtaining-build-logs-with-msbuild?view=vs-2017).

## To view the build log file for a C++ project

1. In **Windows Explorer** or **File Explorer**, open the following file: \...\Visual Studio <Version>\Projects\<ProjectName>\<ProjectName>\Debug\<ProjectName>.txt

## To create a build log file for a managed-code project

1. On the menu bar, choose **Build** > **Build Solution**.
2. In the **Output** window, highlight the information from the build, and then copy it to the **Clipboard**.
3. Open a text editor, such as **Notepad**, paste the information into the file, and then save it.

## To change the amount of information included in the build log

1. On the menu bar, choose **Tools** > **Options**.
2. On the **Projects and Solutions** page, choose the **Build and Run** page.
3. In the **MSBuild project build output verbosity** list, choose one of the following values, and then choose the **OK** button.

| **Verbosity level** | **Description** |
| --- | --- |
| **Quiet** | Displays a summary of the build only. |
| **Minimal** | Displays a summary of the build and errors, warnings, and messages that are categorized as highly important. |
| **Normal** | Displays a summary of the build; errors, warnings, and messages that are categorized as highly important; and the main steps of the build. You'll use this level of detail most frequently. |
| **Detailed** | Displays a summary of the build; errors, warnings, and messages that are categorized as highly important; all of the steps of the build; and messages that are categorized as of normal importance. |
| **Diagnostic** | Displays all data that's available for the build. You can use this level of detail to help debug issues with custom build scripts and other build issues. |

1. For more information, see [Options dialog box, Projects and Solutions, Build and Run](https://docs.microsoft.com/en-us/visualstudio/ide/reference/options-dialog-box-projects-and-solutions-build-and-run?view=vs-2017) and [LoggerVerbosity](https://docs.microsoft.com/en-us/dotnet/api/microsoft.build.framework.loggerverbosity).
2. **Important**
3. You must rebuild the project for your changes to take effect in the **Output** window (all projects) and the .txt file (C++ projects only).