

Contents

[Visual Studio IDE documentation](#)

[Overview](#)

[About Visual Studio](#)

[About the code editor](#)

[About projects and solutions](#)

[More Visual Studio features](#)

[Installation](#)

[Install Visual Studio](#)

[Install offline](#)

[Create an offline installation of Visual Studio](#)

[Install required certificates for offline installation](#)

[Install Visual Studio versions side-by-side](#)

[Select installation locations](#)

[Import or export installation configurations](#)

[Troubleshoot installation issues](#)

[Update Visual Studio](#)

[Modify Visual Studio](#)

[Repair Visual Studio](#)

[Uninstall Visual Studio](#)

[Visual Studio administrator guide](#)

[Overview](#)

[Use the command line](#)

[Use command-line parameters to install Visual Studio](#)

[Command-line parameter examples](#)

[Install on a network](#)

[Create a network-based installation of Visual Studio](#)

[Network considerations](#)

[Install and use Visual Studio behind a firewall or proxy server](#)

[Troubleshoot network errors when you install or use Visual Studio](#)

[Deploy in an enterprise](#)

- [Automate Visual Studio installation with a response file](#)
- [Automatically apply product keys when deploying Visual Studio](#)
- [Set defaults for enterprise deployments of Visual Studio](#)

[Deploy Help Viewer](#)

- [Help Viewer Administrator Guide](#)
- [Command-Line Arguments for the Help Content Manager](#)
- [Help Content Manager Overrides](#)

[Manage & update installations](#)

- [Tools for detecting and managing Visual Studio instances](#)
- [Update a network-based installation of Visual Studio](#)
- [Update Visual Studio while on a servicing baseline](#)
- [Control updates to Visual Studio deployments](#)
- [Manage package payload changes](#)
- [Disable or move the package cache](#)

[Manage subscriptions](#)

- [Visual Studio subscriptions: Administrator overview](#)

[Reference](#)

[Visual Studio workload and component IDs](#)

- [Overview](#)
- [Visual Studio Enterprise](#)
- [Visual Studio Professional](#)
- [Visual Studio Community](#)
- [Visual Studio Team Explorer](#)
- [Visual Studio Desktop Express](#)
- [Visual Studio Build Tools](#)
- [Visual Studio Test Agent](#)
- [Visual Studio Test Controller](#)
- [Visual Studio Test Professional](#)
- [Visual Studio Feedback Client](#)

[Visual Studio build numbers and release dates](#)

[Use Visual Studio from an Azure virtual machine](#)

Containers

[Install Build Tools into a Container](#)

[Advanced Example for Containers](#)

[Known Issues for Containers](#)

Install Help Viewer

Quickstarts

[Visual Studio orientation](#)

[C++: Create a console app](#)

[Python: Create a web app](#)

[Node.js: Create a web app](#)

[F#: Create a web service](#)

[C#: Create a web app](#)

[C#: Create a console app](#)

[Visual Basic: Create a console app](#)

Tutorials

[C#](#)

[F#](#)

[Visual Basic](#)

[C++](#)

[Python](#)

[Node.js](#)

Develop

[Organize and edit code](#)

[Move around in the IDE](#)

[Solutions and projects](#)

[Work with solutions and projects](#)

[Create a new project](#)

[Create solutions and projects](#)

[Open a project from a repo](#)

[Port, migrate, and upgrade projects](#)

[Manage project and solution properties](#)

[Project references](#)

- [Manage references in a project](#)
 - [Add or remove references by using the Reference Manager](#)
 - [Add or remove imported namespaces \(Visual Basic\)](#)
 - [Troubleshoot broken references](#)
- [Manage application resources \(.NET\)](#)
 - [Application settings \(.NET\)](#)
 - [Manage application settings \(.NET\)](#)
 - [Add an app config file to a C# project](#)
- [Sign manifests](#)
 - [Manage assembly and manifest signing](#)
 - [Sign application and deployment manifests](#)
- [Specify an application icon \(Visual Basic, C#\)](#)
- [Target a framework](#)
- [Create templates](#)
 - [Create project and item templates](#)
 - [Add tags to a template](#)
 - [Create project templates](#)
 - [Create multi-project templates](#)
 - [Create item templates](#)
 - [Create multi-file item templates](#)
 - [Create web templates](#)
 - [Troubleshoot templates](#)
- [Locate project and item templates](#)
- [Customize templates](#)
 - [Customize project and item templates](#)
 - [Update existing templates](#)
 - [Substitute parameters in a template](#)
 - [Template parameters](#)
- [64-bit support](#)
- [Develop without projects or solutions](#)
 - [How to: Develop without a project](#)
 - [Customize build and debug tasks](#)

[Connect to a project or repo](#)

[Editor](#)

[Use the editor](#)

[Find and replace](#)

[Find and replace text](#)

[Use regular expressions](#)

[Find-Command box](#)

[Find in Files](#)

[Replace in Files](#)

[Encodings](#)

[Encodings and line breaks](#)

[Save and open files with encoding](#)

[Outlining](#)

[Code generation and refactoring](#)

[Generate and fix code](#)

[Code snippets](#)

[Use code snippets](#)

[C# code snippets reference](#)

[C++ code snippets reference](#)

[Insert XML comments](#)

[Use surround-with code snippets](#)

[Best practices](#)

[Create code snippets](#)

[Walkthrough: Create a code snippet](#)

[Distribute code snippets](#)

[Code snippet functions](#)

[Code snippets schema reference](#)

[Troubleshoot snippets](#)

[Quick Actions](#)

[Common Quick Actions](#)

[Generate class/type](#)

[Generate method](#)

- [Generate field/property/local](#)
- [Generate constructor](#)
- [Generate deconstructor](#)
- [Add parameter to method](#)
- [Generate override](#)
- [Generate Equals and GetHashCode method overrides](#)
- [Generate usings](#)
- [Implement abstract class](#)
- [Implement interface](#)
- [Introduce local variable](#)
- [Refactor code](#)
- [Overview](#)
- [Change method signature](#)
- [Convert anonymous type to class](#)
- [Convert anonymous type to tuple](#)
- [Convert between for loop and foreach statement](#)
- [Convert between Get method and property](#)
- [Convert local function to method](#)
- [Convert foreach loop to LINQ](#)
- [Convert LINQ query to foreach statement](#)
- [Convert switch statement to switch expression](#)
- [Encapsulate field](#)
- [Extract interface](#)
- [Extract method](#)
- [Generate parameter](#)
- [Inline temporary variable](#)
- [IntelliSense completion unimported types](#)
- [Invert conditional expressions and logical operations](#)
- [Invert if statement](#)
- [Move declaration near reference](#)
- [Move type to matching file](#)
- [Move type to namespace](#)

- Pull member up
- Regex completion through IntelliSense
- Remove unreachable code
- Rename
- Sort usings
- Split or merge if statements
- Sync namespace and folder name
- Synchronize type and filename
- Use explicit type
- Use lambda expression or block body
- Unused value assignments, variables, and parameters
- Wrap and align call chains
- Wrap, indent, and align parameters

Walkthrough: Generate code from usage

IntelliSense

- Use IntelliSense
- Visual Basic IntelliSense
- C# IntelliSense
- JavaScript IntelliSense
- Visual C++ IntelliSense
- Configure a C++ project for IntelliSense

Move around in code

- Navigate your code
- Find references in your code
- View type and member definitions
 - Go To Definition and Peek Definition
 - View and edit code by using Peek Definition
- Find code using Go To commands

Customize the editor

- Change text case
- Manage editor modes
- Manage editor windows

- [Change fonts and colors](#)
- [Manage word wrap](#)
- [Display line numbers](#)
- [Display URLs as links](#)
- [Set language-specific editor options](#)
- [Code style](#)
 - [Code style preferences](#)
 - [Use EditorConfig files for code style](#)
 - [.NET coding conventions](#)
 - [Overview](#)
 - [Language conventions](#)
 - [Formatting conventions](#)
 - [Naming conventions](#)
- [Customize the scroll bar](#)
- [Set bookmarks in code](#)
- [CodeLens](#)
 - [Find code history with CodeLens](#)
 - [CodeIndex Command](#)
 - [Editor support for other languages](#)
- [Code views](#)
 - [View the structure of code](#)
 - [Class View and Object Browser icons](#)
- [Use the Task List](#)
- [Class Designer](#)
 - [Design classes in Class Designer](#)
 - [Add class diagrams to projects](#)
 - [Customize class diagrams](#)
 - [Copy class diagram elements to a Microsoft Office document](#)
 - [Export class diagrams as images](#)
 - [Print class diagrams](#)
 - [Add comments to class diagrams](#)
 - [Create classes and types](#)

- [Create types](#)
 - [Create inheritance between types](#)
 - [Create associations between types](#)
 - [Visualize a collection association](#)
 - [Create and configure type members](#)
- [View types and relationships](#)
 - [View existing types](#)
 - [View inheritance between types](#)
 - [Member notation and association notation](#)
- [Refactor classes and types](#)
- [Implement an interface](#)
- [Split a class into partial classes](#)
- [Create a nullable type](#)
- [C++ in Class Designer](#)
 - [Work with Visual C++ code](#)
 - [C++ classes](#)
 - [C++ structures](#)
 - [C++ enumerations](#)
 - [C++ typedefs](#)
- [Keyboard and mouse shortcuts](#)
- [Class Designer errors](#)
- [Improve your code](#)
- [Tips and tricks](#)
 - [Productivity features](#)
 - [Productivity shortcuts](#)
 - [Tips for C# developers](#)
 - [Customize keyboard shortcuts](#)
 - [Use the keyboard exclusively](#)
- [Work with...](#)
 - [Windows Forms apps](#)
 - [Windows Forms Designer overview](#)
 - [Get started with Windows Forms Designer](#)

[Arrange controls >>](#)

[Create a data-bound control >>](#)

[Create custom controls >>](#)

[Disable DPI awareness for Windows Forms Designer](#)

[Reference](#)

[System.Windows.Forms API >>](#)

[Tutorials](#)

[Tutorial 1: Create a picture viewer \(C#\)](#)

[Overview](#)

[Step 1: Create a Windows Forms App project](#)

[Step 2: Run your app](#)

[Step 3: Set your form properties](#)

[Step 4: Lay out your form with a TableLayoutPanel control](#)

[Step 5: Add controls to your form](#)

[Step 6: Name your button controls](#)

[Step 7: Add dialog components to your form](#)

[Step 8: Write code for the Show a Picture button event handler](#)

[Step 9: Review, comment, and test your code](#)

[Step 10: Write code for additional buttons and a check box](#)

[Step 11: Run your app and try other features](#)

[Tutorial 2: Create a timed math quiz \(C#\)](#)

[Overview](#)

[Step 1: Create a project and add labels to your form](#)

[Step 2: Create a random addition problem](#)

[Step 3: Add a countdown timer](#)

[Step 4: Add the CheckTheAnswer\(\) method](#)

[Step 5: Add Enter event handlers for the NumericUpDown controls](#)

[Step 6: Add a subtraction problem](#)

[Step 7: Add multiplication and division problems](#)

[Step 8: Customize the quiz](#)

[Tutorial 3: Create a matching game \(C#\)](#)

[Overview](#)

- [Step 1: Create a project and add a table to your form](#)
- [Step 2: Add a random object and a list of icons](#)
- [Step 3: Assign a random icon to each label](#)
- [Step 4: Add a click event handler to each label](#)
- [Step 5: Add label references](#)
- [Step 6: Add a timer](#)
- [Step 7: Keep pairs visible](#)
- [Step 8: Add a method to verify whether the player won](#)
- [Step 9: Try other features](#)

F#

[F# development](#)

[Target older .NET versions](#)

3D graphical assets

[Work with 3D assets for games and apps](#)

[How to: Use 3D assets in your game or app](#)

Textures and images

[Work with textures and images](#)

[Image editor](#)

[Image editor examples](#)

[Create a basic texture](#)

[Create and modify MIP levels](#)

3D models

[Work with 3D models](#)

[Model editor](#)

[Model editor examples](#)

[Create a basic 3D model](#)

[Modify the pivot point of a 3D model](#)

[Model 3D terrain](#)

[Apply a shader to a 3D model](#)

Shaders

[Work with shaders](#)

[Shader designer](#)

[Shader designer nodes](#)

[Overview](#)

[Constant nodes](#)

[Parameter nodes](#)

[Texture nodes](#)

[Math nodes](#)

[Utility nodes](#)

[Filter nodes](#)

[Shader designer examples](#)

[Create a basic color shader](#)

[Create a basic Lambert shader](#)

[Create a basic Phong shader](#)

[Create a basic texture shader](#)

[Create a grayscale texture shader](#)

[Create a geometry-based gradient shader](#)

[Walkthrough: Create a realistic 3D billiard ball](#)

[Export a shader](#)

[Export textures](#)

[Export a texture that contains mipmaps](#)

[Export a texture that has premultiplied alpha](#)

[Export a texture for use with Direct2D or JavaScript apps](#)

[XAML files >>](#)

[XML and XSLT files >>](#)

[Docker containers >>](#)

[Workflows >>](#)

[Mobile apps](#)

[Office and SharePoint apps >>](#)

[Access data >>](#)

[Develop inclusive apps](#)

[Design accessible apps](#)

[Design international apps](#)

[Build](#)

[Overview](#)

[Walkthrough: Build an application](#)

[Build and clean projects and solutions](#)

[Overview](#)

[Change the build output directory](#)

[Build to a common output directory](#)

[Specify custom build events](#)

[Set multiple startup projects](#)

[Create and remove project dependencies](#)

[View, save, and configure build log files](#)

[Exclude projects from a build](#)

[Suppress compiler warnings](#)

[Build actions](#)

[Build configurations](#)

[Understand build configurations](#)

[Create and edit configurations](#)

[Manage build configurations with Visual Basic developer settings](#)

[Build multiple configurations simultaneously](#)

[Build platforms](#)

[Understand build platforms](#)

[Configure projects to target a platform](#)

[Configure projects to target multiple platforms](#)

[MSBuild >>](#)

[Azure Pipelines and TFS >>](#)

[Specify build events \(Visual Basic\)](#)

[Specify build events \(C#\)](#)

[Configure warnings in Visual Basic](#)

[Walkthrough: Create a multiple-computer build environment](#)

[Debug >>](#)

[Test >>](#)

[Measure performance >>](#)

[Analyze code quality >>](#)

[Deploy >>](#)

[Extend Visual Studio >>](#)

[Analyze and Model Architecture >>](#)

[Personalize Visual Studio](#)

[Customize the IDE](#)

[Quickstart: Personalize theme and text colors](#)

[Environment settings](#)

[Synchronized settings](#)

[Fonts and colors](#)

[Menus and toolbars](#)

[Window layouts](#)

[File nesting in Solution Explorer](#)

[Startup behavior](#)

[Manage extensions](#)

[Manage external tools](#)

[Personalization accounts](#)

[Sign in to Visual Studio](#)

[Work with multiple user accounts](#)

[Extend a trial version or update a license](#)

[Tuning the performance of Visual Studio](#)

[Optimize performance](#)

[Startup time](#)

[Load a filtered solution](#)

[Tips and tricks](#)

[Accessibility](#)

[Manage accessibility features](#)

[Set IDE accessibility options](#)

[Accessibility tips and tricks](#)

[Accessibility products and services from Microsoft](#)

[Reference](#)

[Keyboard shortcuts](#)

[Popular commands](#)

[All commands](#)

Visual Studio commands

[Reference](#)

[Command aliases](#)

[Add Existing Item command](#)

[Add Existing Project command](#)

[Add New Item command](#)

[Alias command](#)

[Evaluate Statement command](#)

[Find command](#)

[Find in Files command](#)

[Go To command](#)

[Import and Export Settings command](#)

[List Call Stack command](#)

[List Disassembly command](#)

[List Memory command](#)

[List Modules command](#)

[List Registers command](#)

[List Source command](#)

[List Threads command](#)

[Log Command window output command](#)

[New File command](#)

[Open File command](#)

[Open Project command](#)

[Print command](#)

[Quick Watch command](#)

[Replace command](#)

[Replace In Files command](#)

[Set Current Process](#)

[Set Current Stack Frame command](#)

[Set Current Thread command](#)

[Set Radix command](#)

[Shell command](#)

ShowWebBrowser command

Start command

Symbol Path command

Toggle Breakpoint command

Watch command

General user interface elements

Call Hierarchy

Preview Changes

Choose Toolbox items, WPF components

Code snippet picker

Command window

Convert dialog box

Error List window

File Properties, JavaScript

Go To Line

Immediate window

Miscellaneous files

Options

Options dialog box

Environment

General

Accounts

AutoRecover

Documents

Extensions

Find and Replace

Fonts and Colors

Import and Export Settings

International Settings

Keyboard

Preview Features

Notifications

[Quick Launch](#)

[Startup](#)

[Tabs and Windows](#)

[Task List](#)

[Trust Settings](#)

[Web Browser](#)

[Projects and Solutions](#)

[General](#)

[Build and Run](#)

[Locations](#)

[VB Defaults](#)

[VC++ Project Settings](#)

[Web Projects](#)

[Text Editor](#)

[General](#)

[File Extension](#)

[All Languages](#)

[General](#)

[Scroll Bars](#)

[Tabs](#)

[Basic](#)

[Advanced](#)

[Code Style](#)

[IntelliSense](#)

[C#](#)

[Advanced](#)

[Code Style > Formatting](#)

[IntelliSense](#)

[C/C++](#)

[Advanced](#)

[Experimental](#)

[Formatting](#)

[View](#)

[F#](#)

[Advanced](#)

[Code Fixes](#)

[CodeLens](#)

[IntelliSense](#)

[HTML \(Web Forms\)](#)

[Formatting](#)

[Miscellaneous](#)

[Validation](#)

[JavaScript/TypeScript](#)

[Code Validation](#)

[Formatting](#)

[IntelliSense](#)

[Linting](#)

[Project](#)

[U-SQL](#)

[Formatting](#)

[IntelliSense](#)

[XAML](#)

[Formatting](#)

[Miscellaneous](#)

[XML](#)

[Formatting](#)

[Miscellaneous](#)

[Windows Forms Designer](#)

[General](#)

[Data UI Customization](#)

[XAML Designer](#)

[Output window](#)

[Project Properties](#)

[Reference](#)

[Application Page, Project Designer \(UWP\)](#)

[Application Page, Project Designer \(Visual Basic\)](#)

[Application Page](#)

[Assembly Information Dialog Box](#)

[Application Page, Project Designer \(C#\)](#)

[Build Events Page, Project Designer \(C#\)](#)

[Build Events Page](#)

[Pre-build Event-Post-build Event Command Line Dialog Box](#)

[Build Page, Project Designer \(C#\)](#)

[Build Page](#)

[Advanced Build Settings Dialog Box \(C#\)](#)

[Code Analysis, Project Designer](#)

[Compile Page, Project Designer \(Visual Basic\)](#)

[Compile Page](#)

[Advanced Compiler Settings Dialog Box \(Visual Basic\)](#)

[Build Events Dialog Box \(Visual Basic\)](#)

[Debug Page, Project Designer](#)

[My Extensions Page, Project Designer \(Visual Basic\)](#)

[Publish Page, Project Designer](#)

[Publish Page](#)

[Prerequisites Dialog Box](#)

[References Page, Project Designer \(Visual Basic\)](#)

[Security Page, Project Designer](#)

[Security Page](#)

[Advanced Security Settings Dialog Box](#)

[Services Page, Project Designer](#)

[Services Page](#)

[Advanced Settings for Services Dialog Box](#)

[Settings Page, Project Designer](#)

[Signing Page, Project Designer](#)

[Property Pages, JavaScript](#)

[Properties window](#)

[Team Explorer reference](#)

[Toolbox](#)

[Reference](#)

[Toolbox, Components Tab](#)

[Toolbox, Data Tab](#)

[Toolbox, HTML Tab](#)

[Devenv command-line switches](#)

[Reference](#)

[-? \(devenv.exe\)](#)

[-Build \(devenv.exe\)](#)

[-Clean \(devenv.exe\)](#)

[-Command \(devenv.exe\)](#)

[-DebugExe \(devenv.exe\)](#)

[-Deploy \(devenv.exe\)](#)

[-Diff \(devenv.exe\)](#)

[-DoNotLoadProjects \(devenv.exe\)](#)

[-Edit \(devenv.exe\)](#)

[-LCID \(devenv.exe\)](#)

[-Log \(devenv.exe\)](#)

[-NoSplash \(devenv.exe\)](#)

[-Out \(devenv.exe\)](#)

[-Project \(devenv.exe\)](#)

[-ProjectConfig \(devenv.exe\)](#)

[-Rebuild \(devenv.exe\)](#)

[-ResetSettings \(devenv.exe\)](#)

[-Run \(devenv.exe\)](#)

[-Runexit \(devenv.exe\)](#)

[-SafeMode \(devenv.exe\)](#)

[-Upgrade \(devenv.exe\)](#)

[-UseEnv \(devenv.exe\)](#)

[Security](#)

[Develop secure applications](#)

[Run Visual Studio as normal user or administrator](#)

[Windows Information Protection \(WIP\)](#)

[Bidirectional language input](#)

[Microsoft Help Viewer](#)

[Overview](#)

[Install and Manage Local Content](#)

[Find topics in Help Viewer](#)

[Find Topics in the Index](#)

[Find Topics in the Table of Contents](#)

[Search](#)

[Search for Topics](#)

[Logical and Advanced Operators in Search Expressions](#)

[Customize Help Viewer](#)

[Accessibility](#)

[Shortcut Keys](#)

[The Visual Studio image library](#)

[Dotfuscator Community](#)

[Overview](#)

[Capabilities of Dotfuscator](#)

[Install Dotfuscator Community](#)

[Upgrade Dotfuscator Community](#)

[Resources](#)

[What's new in Visual Studio 2017](#)

[What's new in Visual Studio 2019](#)

[Release notes & system requirements](#)

[Release rhythm](#)

[Visual Studio 2017](#)

[Current release notes](#)

[Preview release notes](#)

[Release notes history](#)

[Distributable code](#)

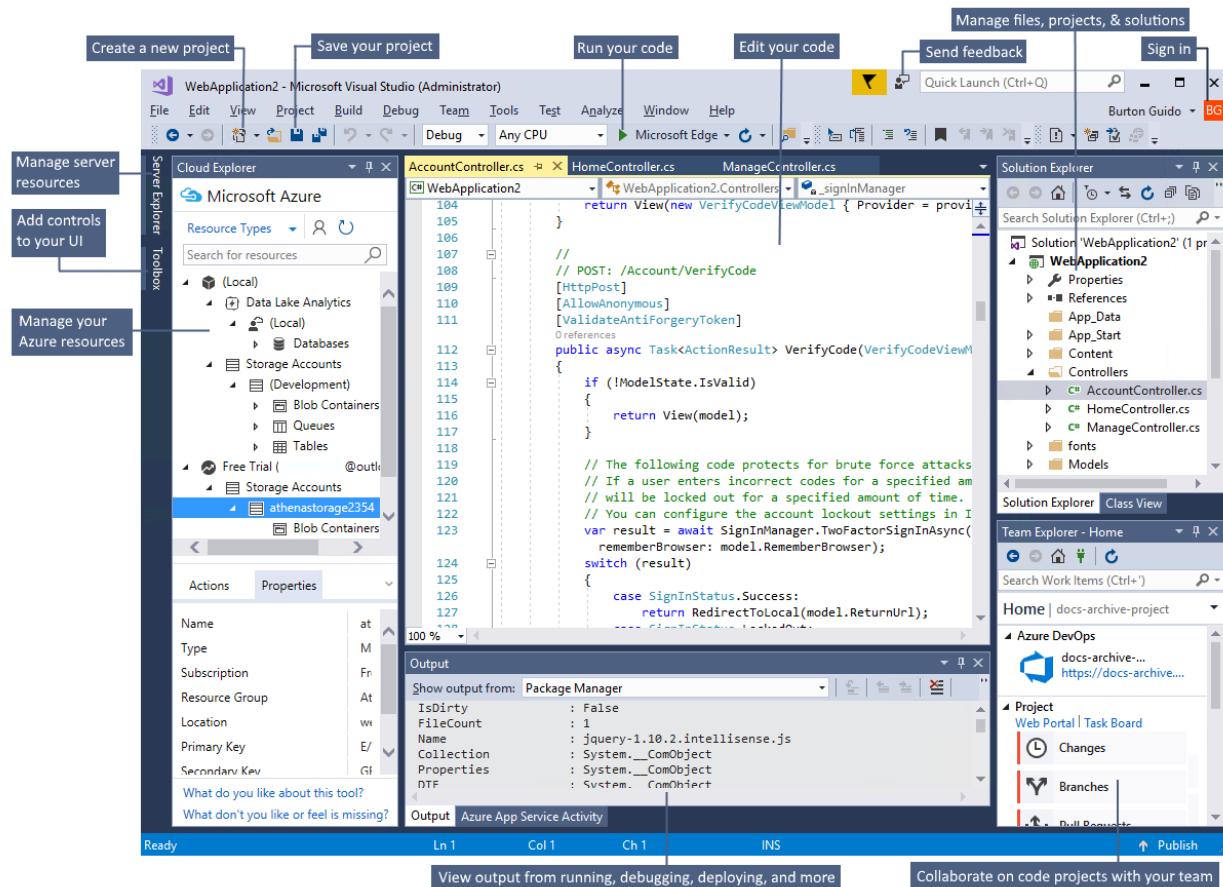
[Platform compatibility](#)

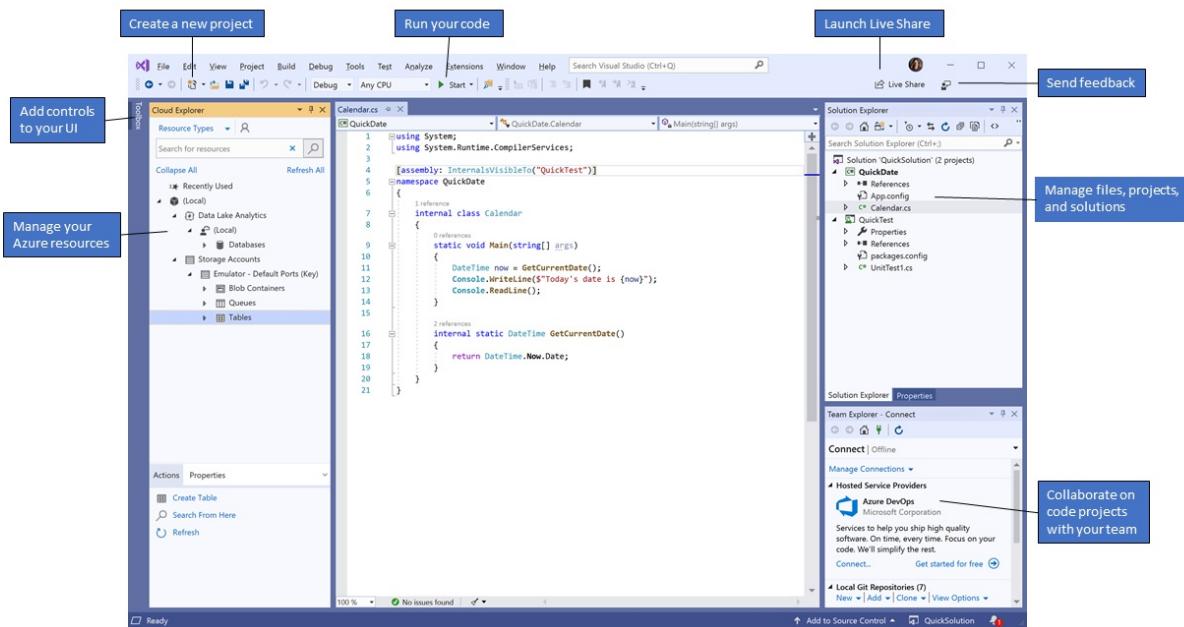
- [System requirements](#)
- [Visual Studio 2019](#)
 - [Current release notes](#)
 - [Preview release notes](#)
 - [Release notes history](#)
 - [Distributable code](#)
 - [Platform compatibility](#)
 - [System requirements](#)
- [License terms](#)
- [Support lifecycle and servicing](#)
- [Customer Experience Improvement Program](#)
 - [Overview](#)
 - [System-generated logs](#)
 - [Resources for troubleshooting IDE errors](#)
- [Send feedback](#)
 - [Overview](#)
 - [Report a problem](#)
 - [How to report a problem](#)
 - [Report a problem states and FAQ](#)
 - [Get performance issues fixed quicker](#)
 - [Create minidumps with callstacks](#)
 - [Create logs for MSBuild problems](#)
 - [Collect an ETL trace with PerfView](#)
 - [Suggest a feature](#)
 - [Developer Community data privacy](#)

Welcome to the Visual Studio IDE

10/18/2019 • 14 minutes to read • [Edit Online](#)

The Visual Studio *integrated development environment* is a creative launching pad that you can use to edit, debug, and build code, and then publish an app. An integrated development environment (IDE) is a feature-rich program that can be used for many aspects of software development. Over and above the standard editor and debugger that most IDEs provide, Visual Studio includes compilers, code completion tools, graphical designers, and many more features to ease the software development process.





This image shows Visual Studio with an open project and several key tool windows you'll likely use:

- **Solution Explorer** (top right) lets you view, navigate, and manage your code files. **Solution Explorer** can help organize your code by grouping the files into [solutions and projects](#).
- The **editor window** (center), where you'll likely spend a majority of your time, displays file contents. This is where you can edit code or design a user interface such as a window with buttons and text boxes.
- The **Output window** (bottom center) is where Visual Studio sends notifications such as debugging and error messages, compiler warnings, publishing status messages, and more. Each message source has its own tab.
- **Team Explorer** (bottom right) lets you track work items and share code with others using version control technologies such as [Git](#) and [Team Foundation Version Control \(TFVC\)](#).

Editions

Visual Studio is available for Windows and Mac. [Visual Studio for Mac](#) has many of the same features as Visual Studio 2017, and is optimized for developing cross-platform and mobile apps. This article focuses on the Windows version of Visual Studio 2017.

There are three editions of Visual Studio 2017: Community, Professional, and Enterprise. See [Compare Visual Studio 2017 IDEs](#) to learn about which features are supported in each edition.

Visual Studio is available for Windows and Mac. [Visual Studio for Mac](#) has many of the same features as Visual Studio 2019, and is optimized for developing cross-platform and mobile apps. This article focuses on the Windows version of Visual Studio 2019.

There are three editions of Visual Studio 2019: Community, Professional, and Enterprise. See [Compare Visual Studio IDEs](#) to learn about which features are supported in each edition.

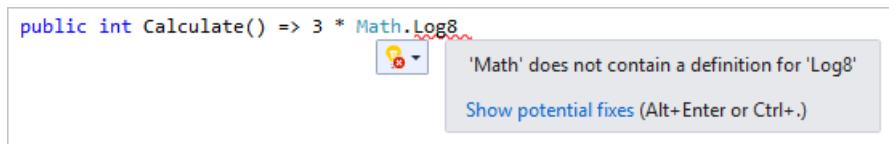
Popular productivity features

Some of the popular features in Visual Studio that help you to be more productive as you develop software include:

- Squiggles and [Quick Actions](#)

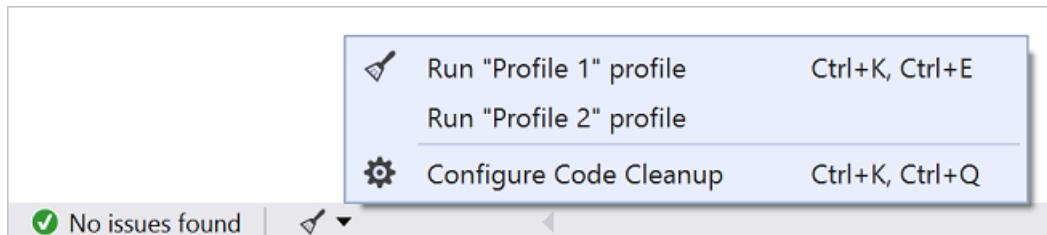
Squiggles are wavy underlines that alert you to errors or potential problems in your code as you type.

These visual clues enable you to fix problems immediately without waiting for the error to be discovered during build or when you run the program. If you hover over a squiggle, you see additional information about the error. A light bulb may also appear in the left margin with actions, known as Quick Actions, to fix the error.



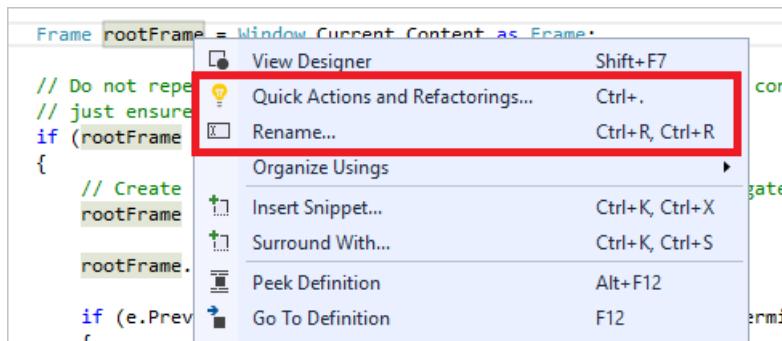
- Code Cleanup

With the click of a button, format your code and apply any code fixes suggested by your [code style settings](#), [.editorconfig conventions](#), and [Roslyn analyzers](#). **Code Cleanup** helps you resolve issues in your code before it goes to code review. (Currently available for C# code only.)



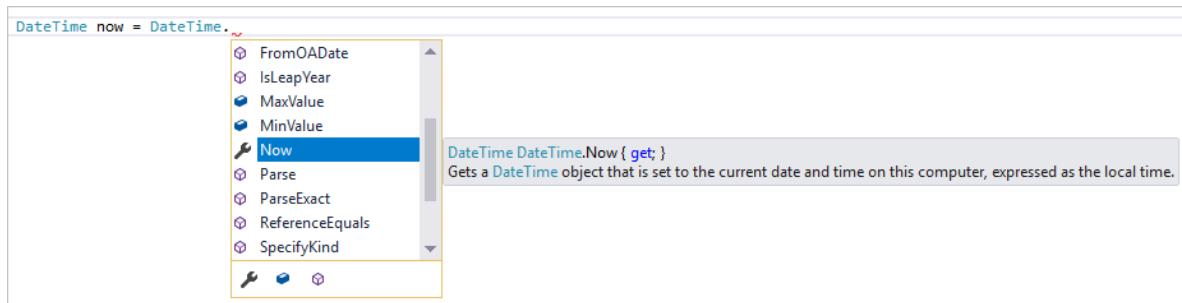
- Refactoring

Refactoring includes operations such as intelligent renaming of variables, extracting one or more lines of code into a new method, changing the order of method parameters, and more.



- IntelliSense

IntelliSense is a term for a set of features that displays information about your code directly in the editor and, in some cases, write small bits of code for you. It's like having basic documentation inline in the editor, which saves you from having to look up type information elsewhere. IntelliSense features vary by language. For more information, see [C# IntelliSense](#), [Visual C++ IntelliSense](#), [JavaScript IntelliSense](#), and [Visual Basic IntelliSense](#). The following illustration shows how IntelliSense displays a member list for a type:

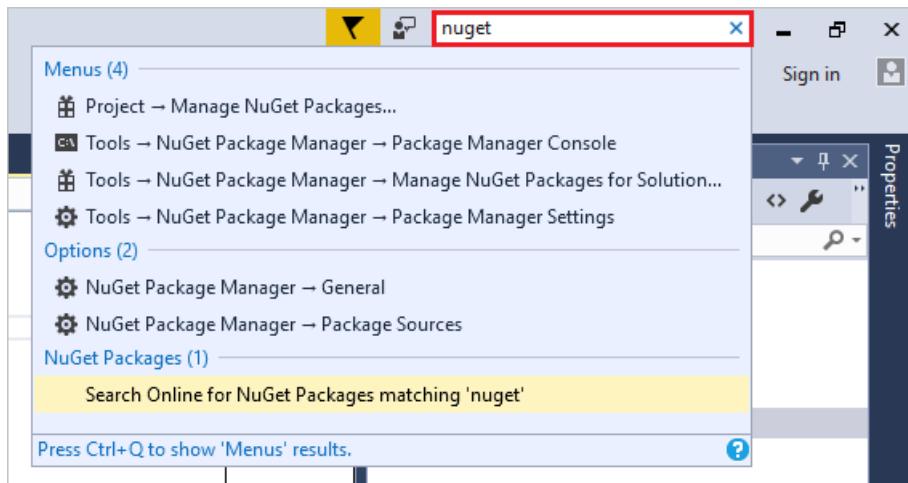


- Search box

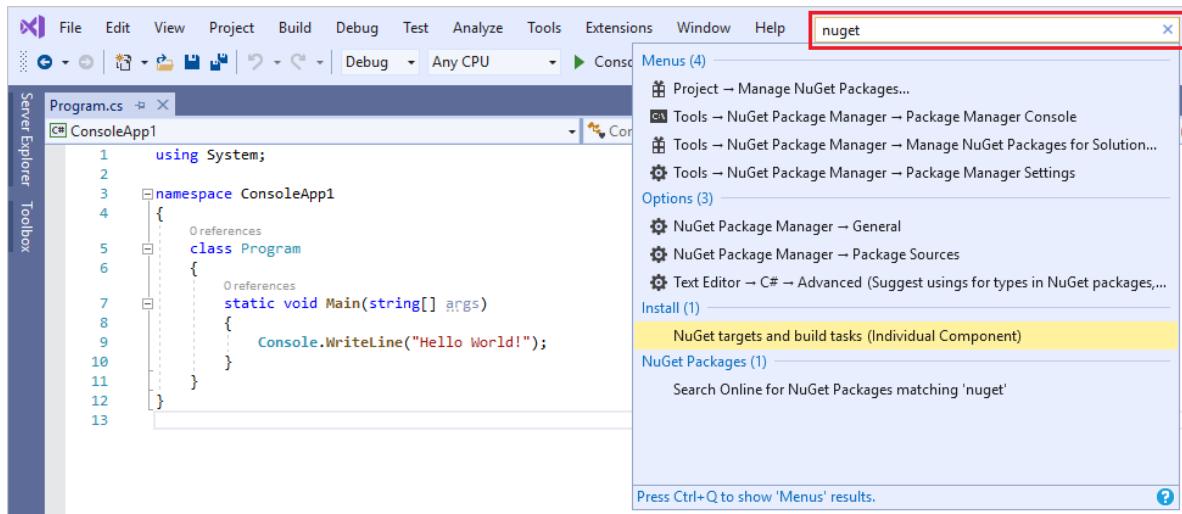
Visual Studio can seem overwhelming at times with so many menus, options, and properties. The search box is a great way to rapidly find what you need in Visual Studio. When you start typing the name of something you're looking for, Visual Studio lists results that take you exactly where you need to go. If you need to add functionality to Visual Studio, for example to add support for an additional programming language, the search box provides results that open Visual Studio Installer to install a workload or individual component.

TIP

Press **Ctrl+Q** as a shortcut to the search box.



For more information, see [Quick Launch](#).

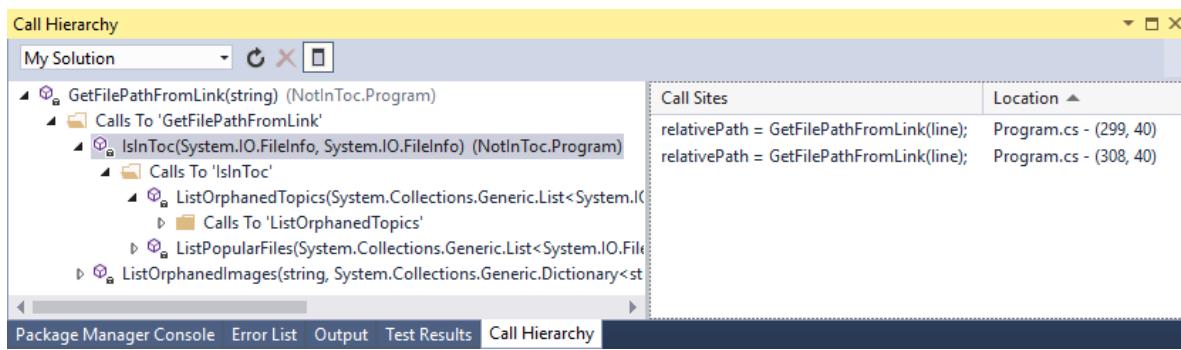


- [Live Share](#)

Collaboratively edit and debug with others in real time, regardless of what your app type or programming language. You can instantly and securely share your project and, as needed, debugging sessions, terminal instances, localhost web apps, voice calls, and more.

- [Call Hierarchy](#)

The **Call Hierarchy** window shows the methods that call a selected method. This can be useful information when you're thinking about changing or removing the method, or when you're trying to track down a bug.



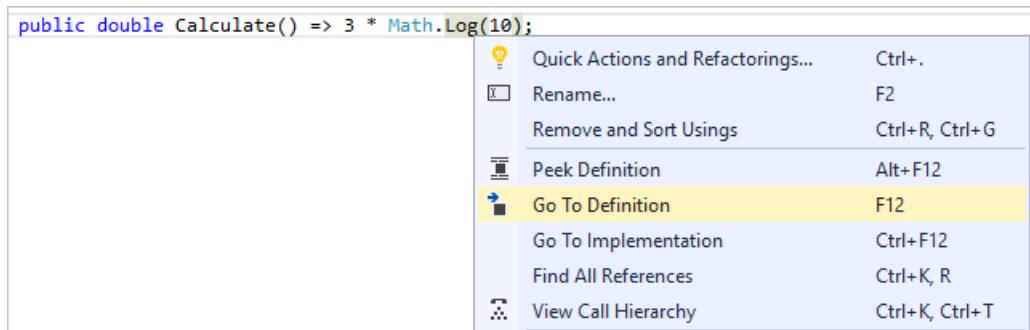
- **CodeLens**

CodeLens helps you find references to your code, changes to your code, linked bugs, work items, code reviews, and unit tests, all without leaving the editor.



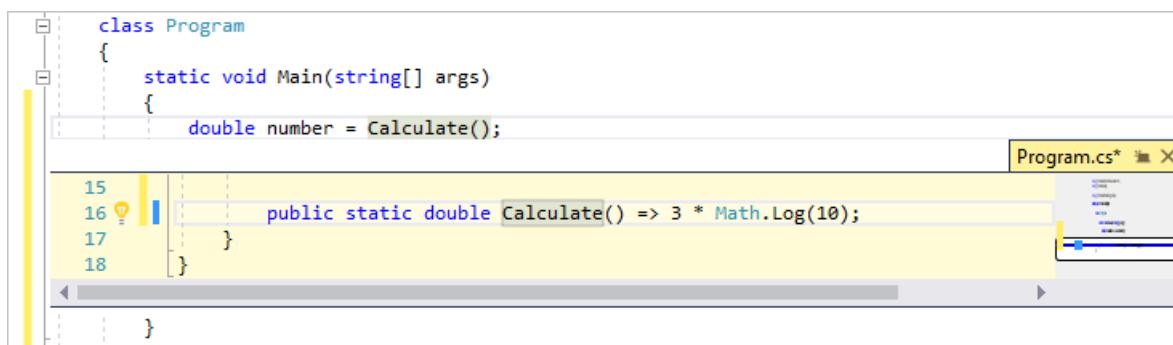
- **Go To Definition**

The Go To Definition feature takes you directly to the location where a function or type is defined.



- **Peek Definition**

The **Peek Definition** window shows the definition of a method or type without actually opening a separate file.



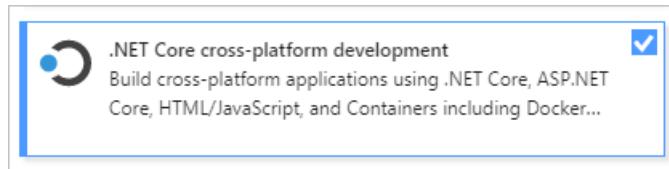
Install the Visual Studio IDE

In this section, you'll create a simple project to try out some of the things you can do with Visual Studio. You'll use

[IntelliSense](#) as a coding aid, debug an app to see the value of a variable during the program's execution, and change the color theme.

To get started, [download Visual Studio](#) and install it on your system. The modular installer enables you to choose and install *workloads*, which are groups of features needed for the programming language or platform you prefer. To follow the steps for [creating a program](#), be sure to select the **.NET Core cross-platform development** workload during installation.

To get started, [download Visual Studio](#) and install it on your system. The modular installer enables you to choose and install *workloads*, which are groups of features needed for the programming language or platform you prefer. To follow the steps for [creating a program](#), be sure to select the **.NET Core cross-platform development** workload during installation.

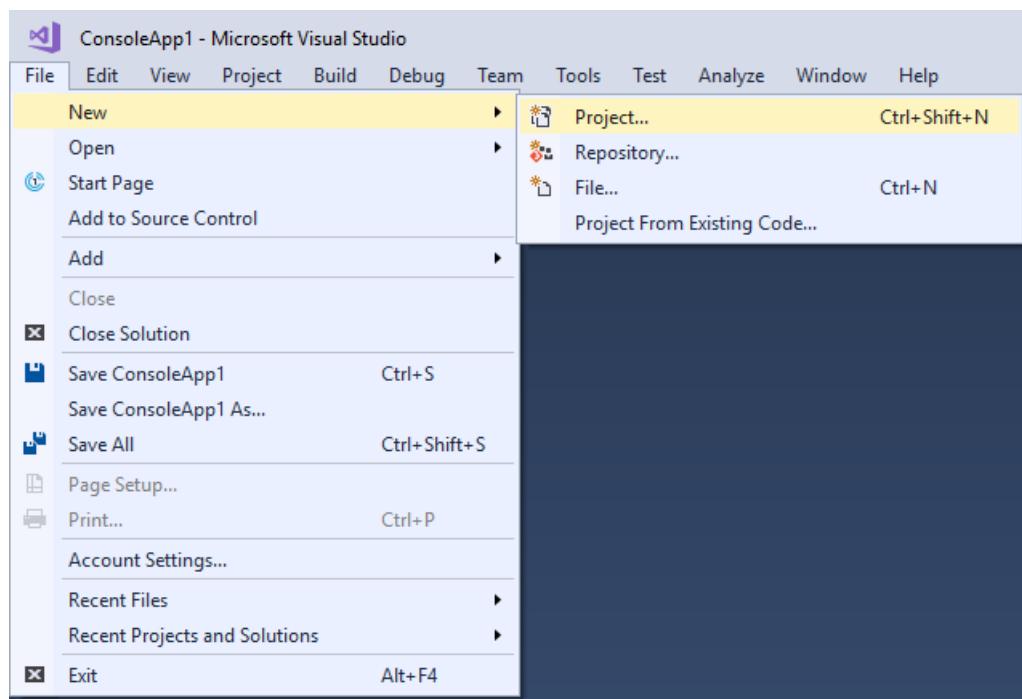


When you open Visual Studio for the first time, you can optionally [sign in](#) using your Microsoft account or your work or school account.

Create a program

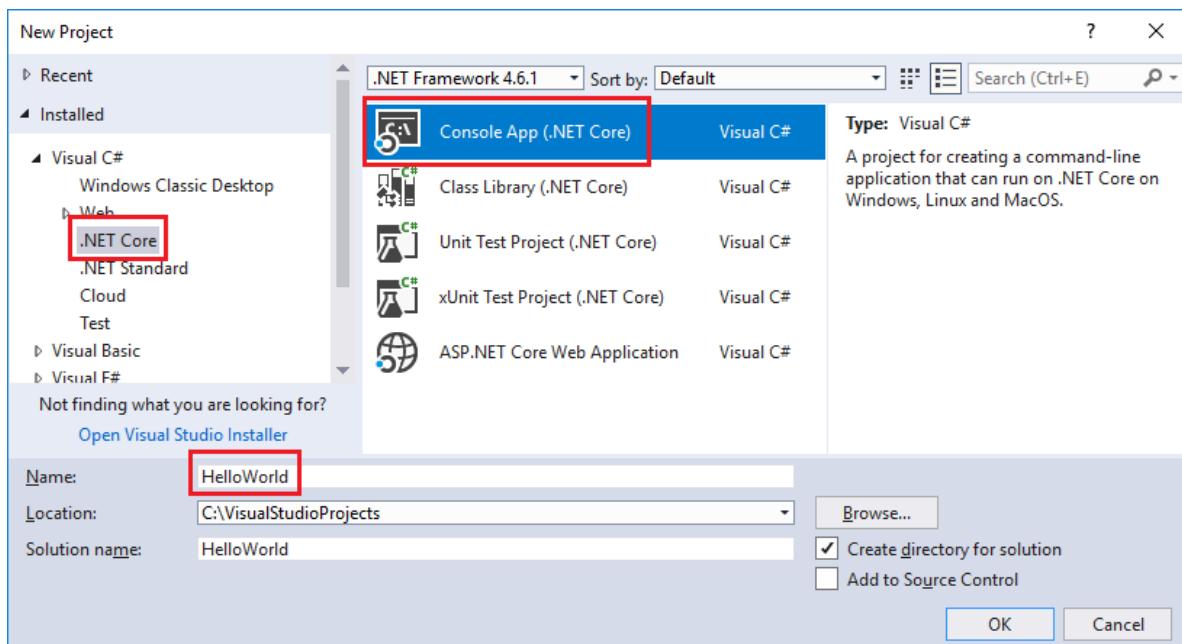
Let's dive in and create a simple program.

1. Open Visual Studio.
2. On the menu bar, choose **File > New > Project**.



The **New Project** dialog box shows several project *templates*. A template contains the basic files and settings needed for a given project type.

3. Choose the **.NET Core** template category under **Visual C#**, and then choose the **Console App (.NET Core)** template. In the **Name** text box, type **HelloWorld**, and then select the **OK** button.

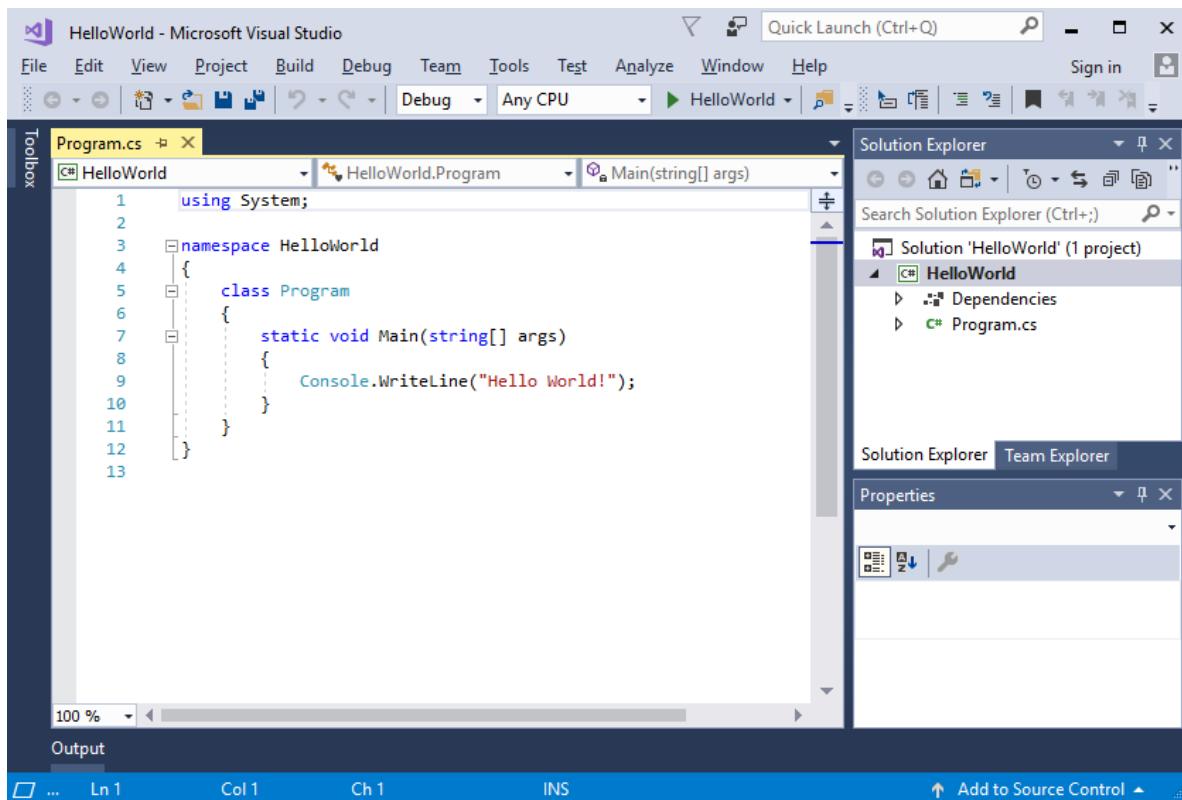


NOTE

If you don't see the **.NET Core** category, you need to install the **.NET Core cross-platform development** workload. To do this, choose the **Open Visual Studio Installer** link on the bottom left of the **New Project** dialog. After Visual Studio Installer opens, scroll down and select the **.NET Core cross-platform development** workload, and then select **Modify**.

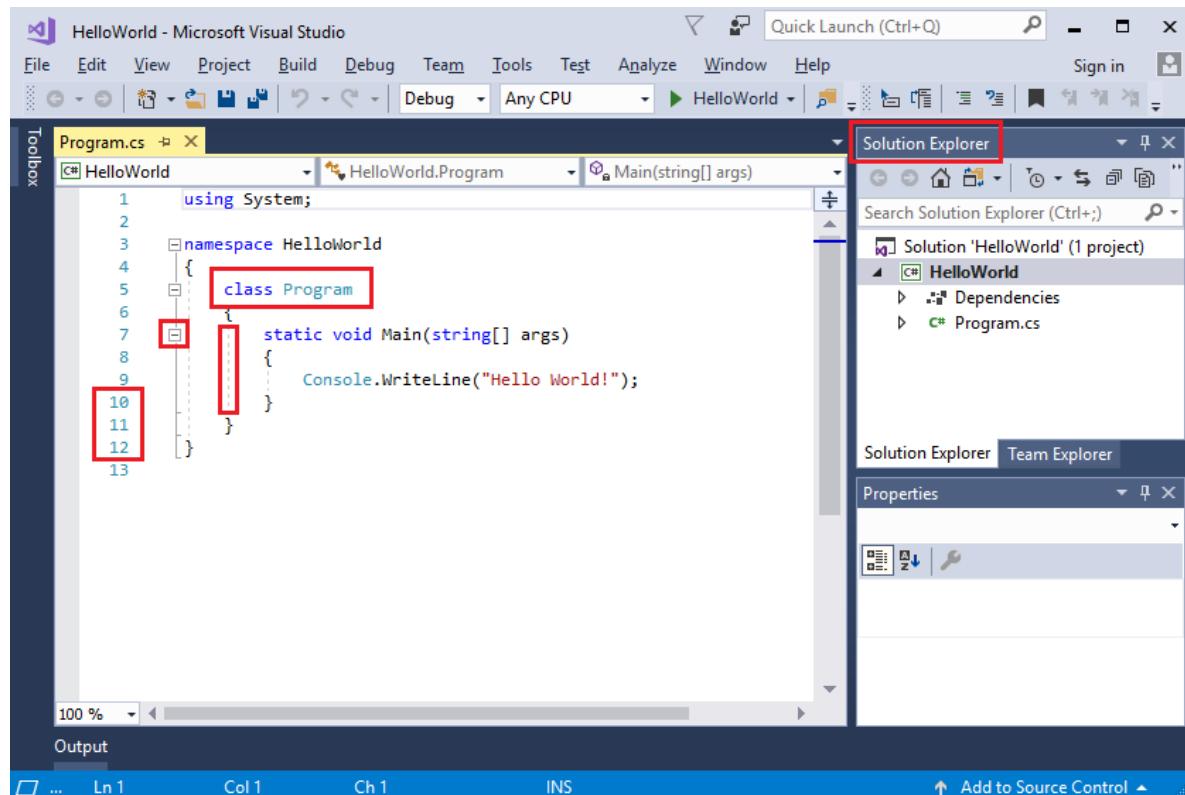
Visual Studio creates the project. It's a simple "Hello World" application that calls the `Console.WriteLine()` method to display the literal string "Hello World!" in the console (program output) window.

Shortly, you should see something like the following:



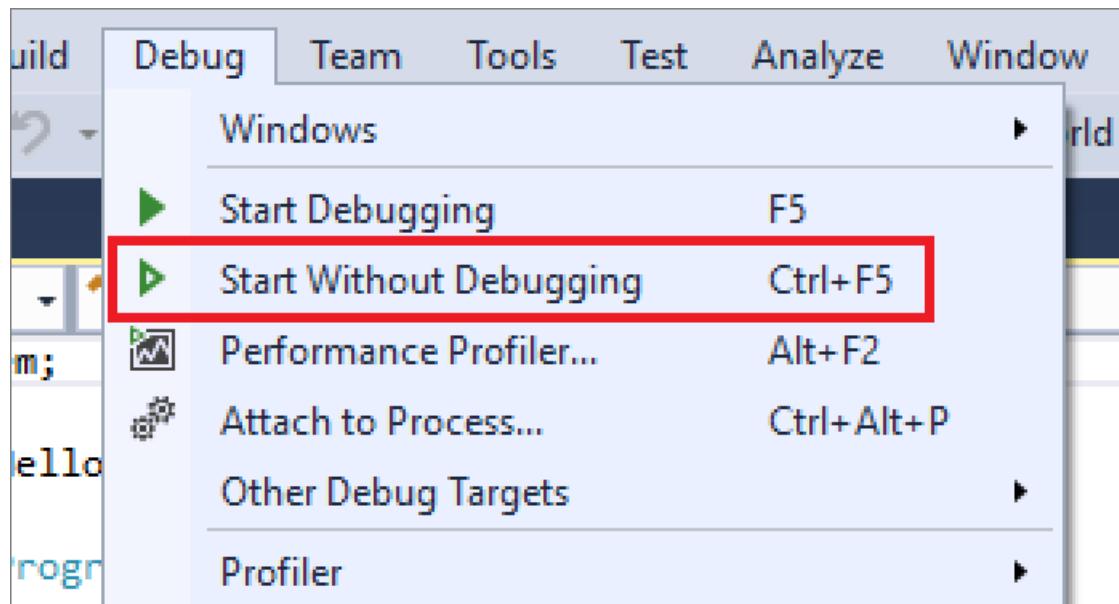
The C# code for your application shows in the editor window, which takes up most of the space. Notice that the text is automatically colorized to indicate different parts of the code, such as keywords and types. In addition, small, vertical dashed lines in the code indicate which braces match one another, and line

numbers help you locate code later. You can choose the small, boxed minus signs to collapse or expand blocks of code. This code outlining feature lets you hide code you don't need, helping to minimize onscreen clutter. The project files are listed on the right side in a window called **Solution Explorer**.

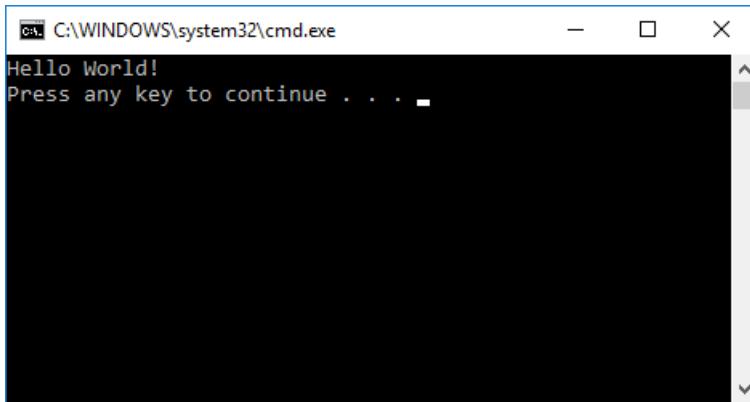


There are other menus and tool windows available, but let's move on for now.

4. Now, start the app. You can do this by choosing **Start Without Debugging** from the **Debug** menu on the menu bar. You can also press **Ctrl+F5**.



Visual Studio builds the app, and a console window opens with the message **Hello World!**. You now have a running app!



```
C:\WINDOWS\system32\cmd.exe
Hello World!
Press any key to continue . . .
```

5. To close the console window, press any key on your keyboard.
6. Let's add some additional code to the app. Add the following C# code before the line that says

```
Console.WriteLine("Hello World!"); :
```

```
Console.WriteLine("\nWhat is your name?");
var name = Console.ReadLine();
```

This code displays **What is your name?** in the console window, and then waits until the user enters some text followed by the **Enter** key.

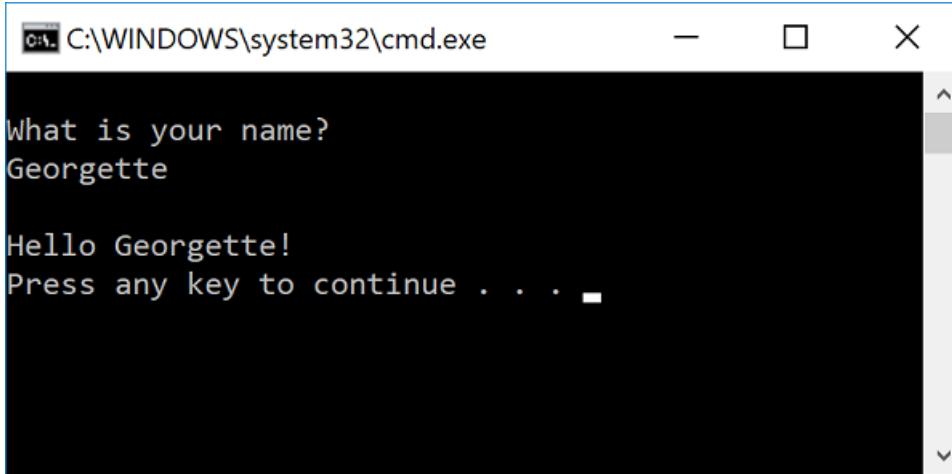
7. Change the line that says `Console.WriteLine("Hello World!");` to the following code:

```
Console.WriteLine($"\\nHello {name}!");
```

8. Run the app again by selecting **Debug > Start Without Debugging** or by pressing **Ctrl+F5**.

Visual Studio rebuilds the app, and a console window opens and prompts you for your name.

9. Enter your name in the console window and press **Enter**.



```
C:\WINDOWS\system32\cmd.exe
What is your name?
Georgette

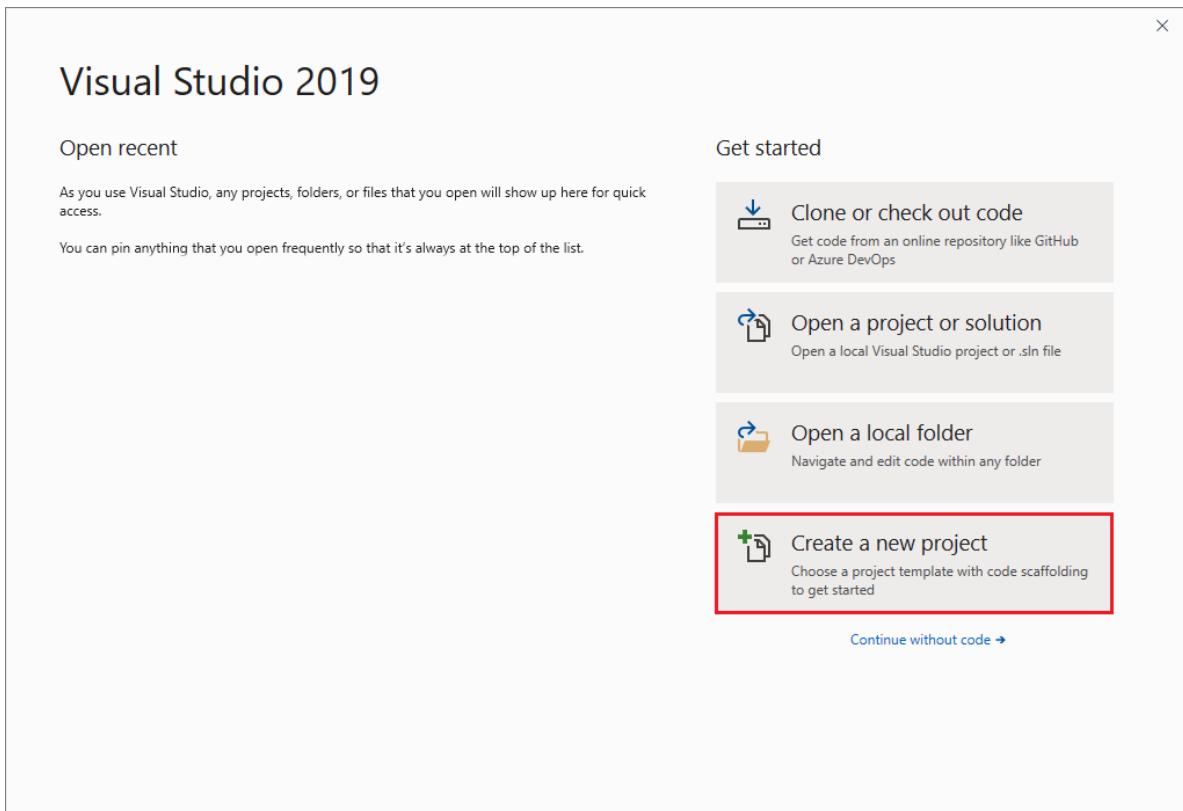
Hello Georgette!
Press any key to continue . . .
```

10. Press any key to close the console window and stop the running program.

1. Open Visual Studio.

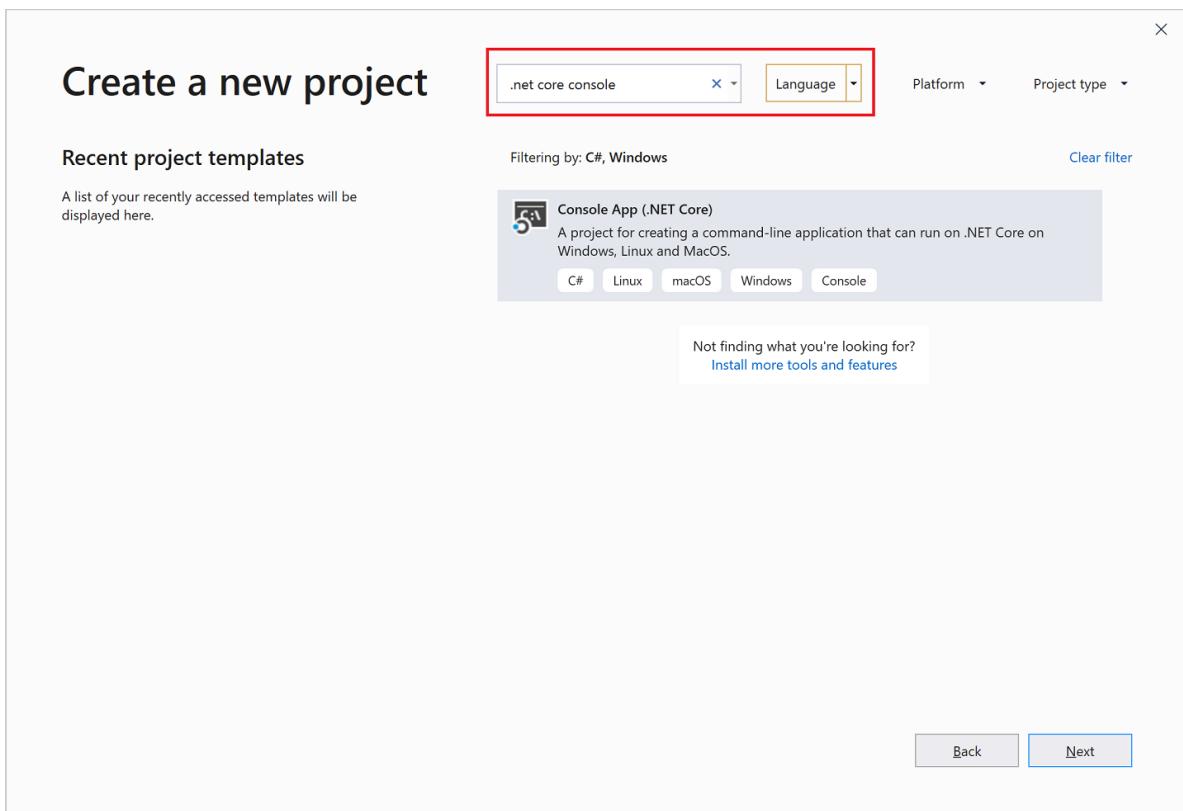
The start window appears with various options for cloning a repo, opening a recent project, or creating a brand new project.

2. Choose **Create a new project**.

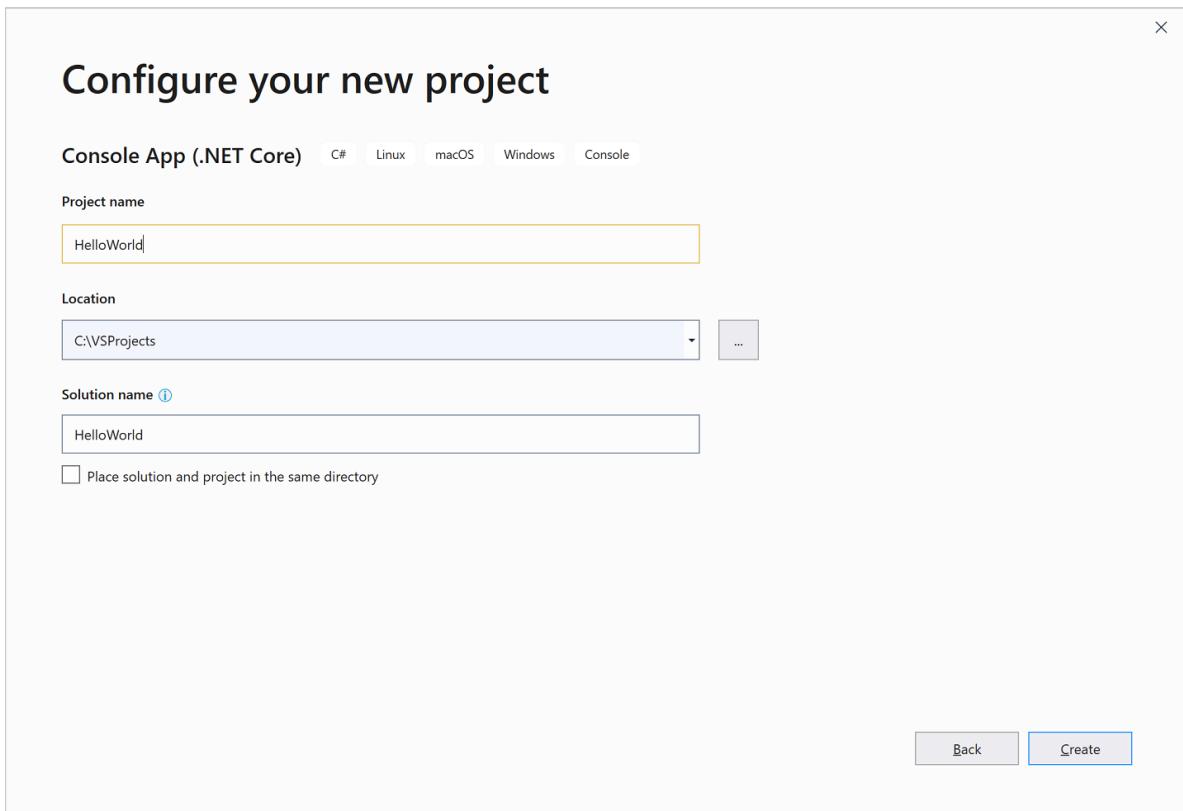


The **Create a new project** window opens and shows several project *templates*. A template contains the basic files and settings needed for a given project type.

3. To find the template we want, type or enter **.net core console** in the search box. The list of available templates is automatically filtered based on the keywords you entered. You can further filter the template results by choosing **C#** from the **Language** drop-down list. Select the **Console App (.NET Core)** template, and then choose **Next**.

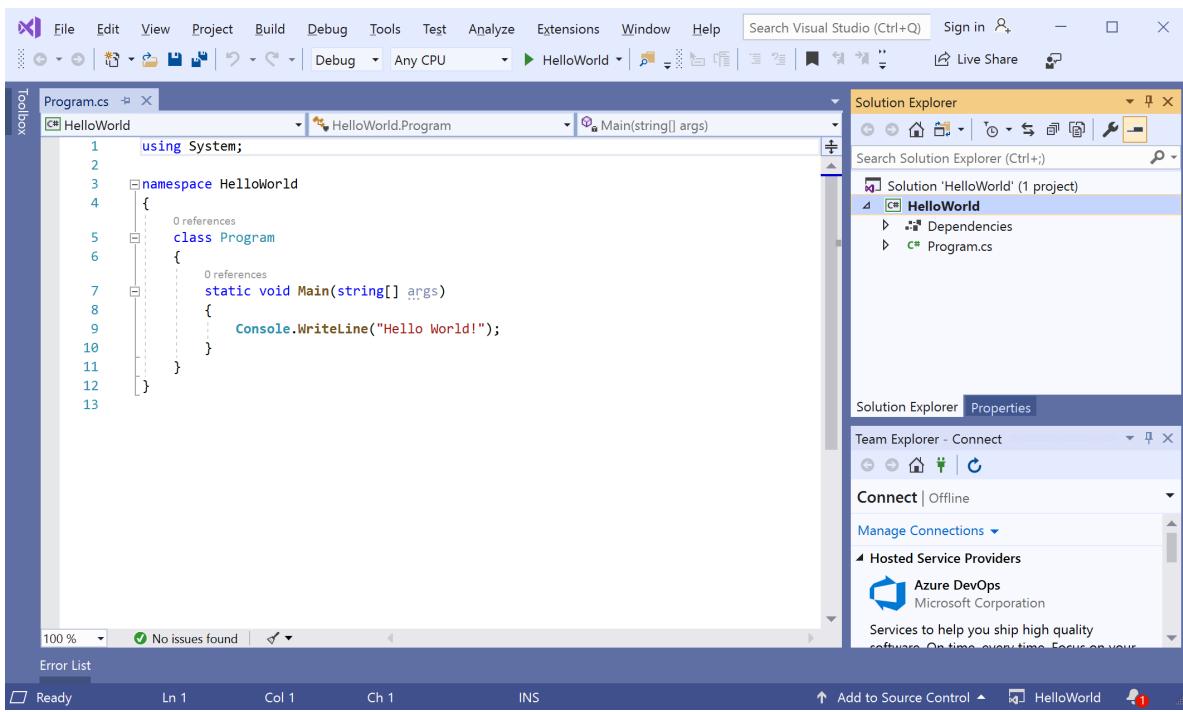


4. In the **Configure your new project** window, enter **HelloWorld** in the **Project name** box, optionally change the directory location for your project files, and then choose **Create**.

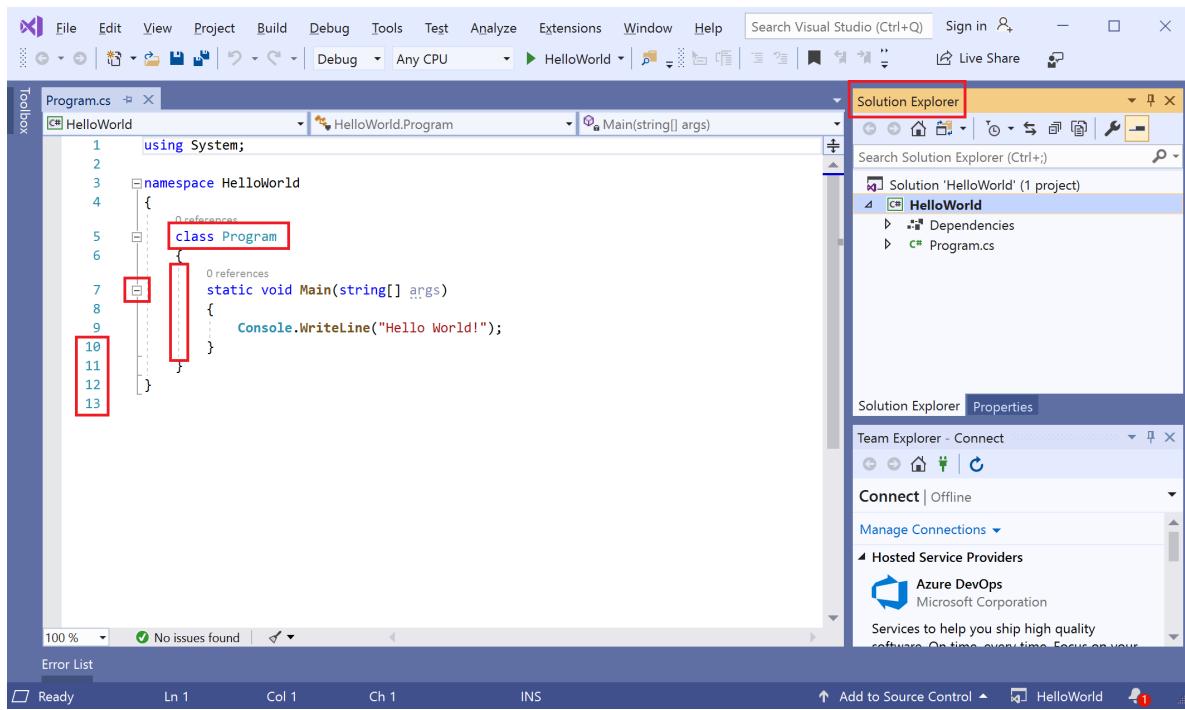


Visual Studio creates the project. It's a simple "Hello World" application that calls the `Console.WriteLine()` method to display the literal string "Hello World!" in the console (program output) window.

Shortly, you should see something like the following:

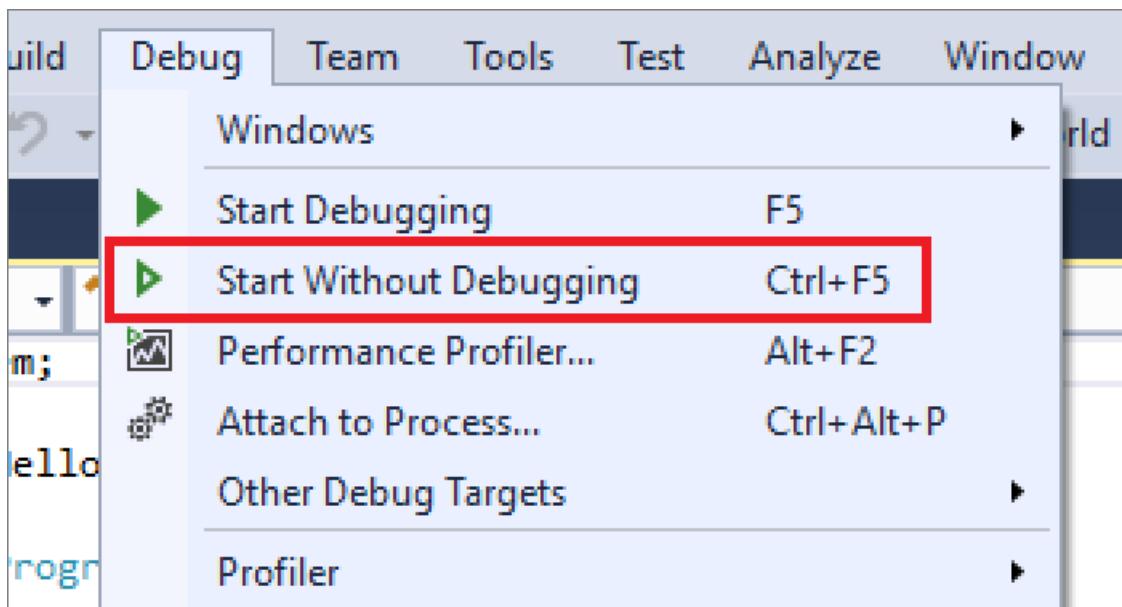


The C# code for your application shows in the editor window, which takes up most of the space. Notice that the text is automatically colorized to indicate different parts of the code, such as keywords and types. In addition, small, vertical dashed lines in the code indicate which braces match one another, and line numbers help you locate code later. You can choose the small, boxed minus signs to collapse or expand blocks of code. This code outlining feature lets you hide code you don't need, helping to minimize onscreen clutter. The project files are listed on the right side in a window called **Solution Explorer**.

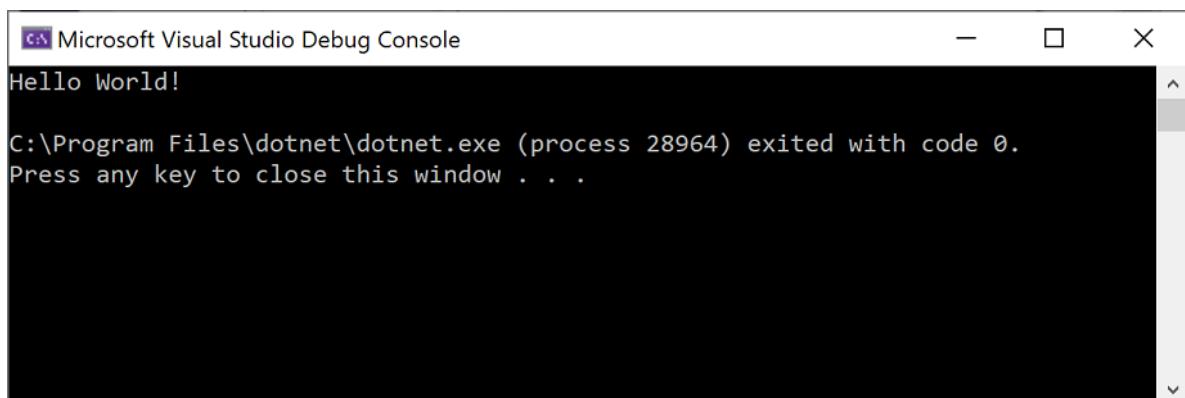


There are other menus and tool windows available, but let's move on for now.

5. Now, start the app. You can do this by choosing **Start Without Debugging** from the **Debug** menu on the menu bar. You can also press **Ctrl+F5**.



Visual Studio builds the app, and a console window opens with the message **Hello World!**. You now have a running app!



- To close the console window, press any key on your keyboard.
- Let's add some additional code to the app. Add the following C# code before the line that says

```
Console.WriteLine("Hello World!"); :
```

```
Console.WriteLine("\nWhat is your name?");
var name = Console.ReadLine();
```

This code displays **What is your name?** in the console window, and then waits until the user enters some text followed by the **Enter** key.

- Change the line that says `Console.WriteLine("Hello World!");` to the following code:

```
Console.WriteLine($"\\nHello {name}!");
```

- Run the app again by selecting **Debug > Start Without Debugging** or by pressing **Ctrl+F5**.

Visual Studio rebuilds the app, and a console window opens and prompts you for your name.

- Enter your name in the console window and press **Enter**.

```
Microsoft Visual Studio Debug Console
What is your name?
Georgette
Hello Georgette!
C:\Program Files\dotnet\dotnet.exe (process 37516) exited with code 0.
Press any key to close this window . . .
```

- Press any key to close the console window and stop the running program.

Use refactoring and IntelliSense

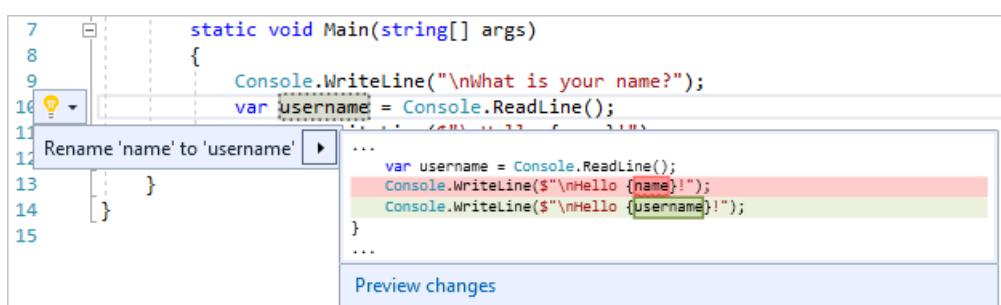
Let's look at a couple of the ways that [refactoring](#) and [IntelliSense](#) can help you code more efficiently.

First, let's rename the `name` variable:

- Double-click the `name` variable to select it.
- Type in the new name for the variable, **username**.

Notice that a gray box appears around the variable, and a light bulb appears in the margin.

- Select the light bulb icon to show the available [Quick Actions](#). Select **Rename 'name' to 'username'**.



The variable is renamed across the project, which in our case is only two places.

```
3  namespace HelloWorld
4  {
5      class Program
6      {
7          static void Main(string[] args)
8          {
9              Console.WriteLine("\nWhat is your name?");
10             var name = Console.ReadLine();
11             Console.WriteLine($"\\nHello {name}!");
12         }
13     }
14 }
```

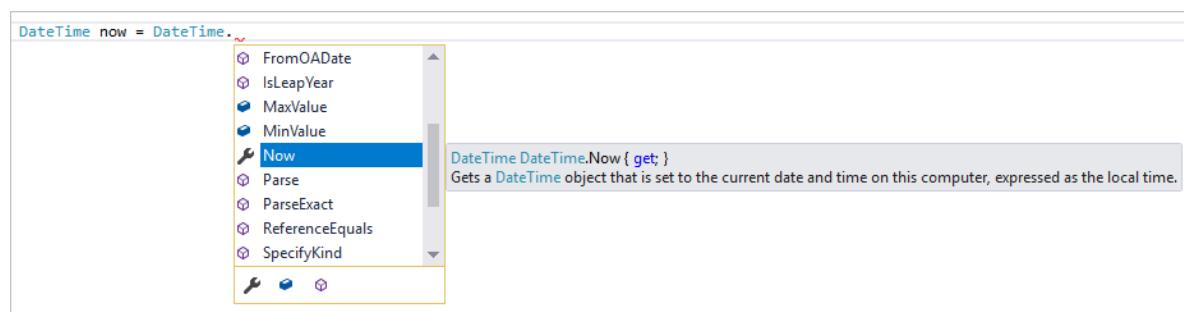
3. Select the light bulb icon to show the available [Quick Actions](#). Select **Rename 'name' to 'username'**.



The variable is renamed across the project, which in our case is only two places.

4. Now let's take a look at IntelliSense. Below the line that says `Console.WriteLine($"\\nHello {username}!");`, type `DateTime now = DateTime.`.

A box displays the members of the `DateTime` class. In addition, the description of the currently selected member displays in a separate box.



5. Select the member named **Now**, which is a property of the class, by double-clicking on it or pressing **Tab**. Complete the line of code by adding a semi-colon to the end.
6. Below that, type in or paste the following lines of code:

```
int dayOfYear = now.DayOfYear;

Console.Write("Day of year: ");
Console.WriteLine(dayOfYear);
```

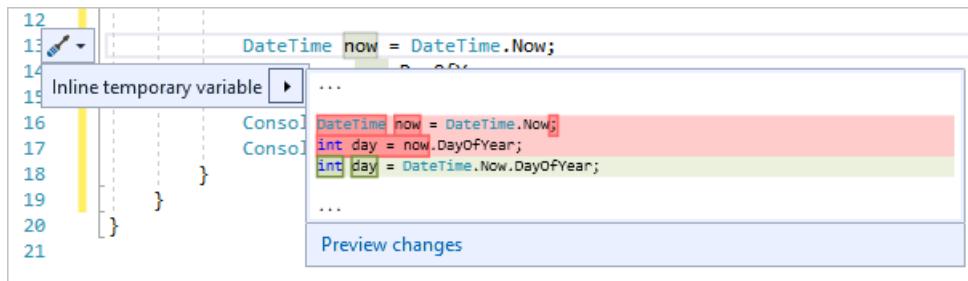
TIP

`Console.Write` is a little different to `Console.WriteLine` in that it doesn't add a line terminator after it prints. That means that the next piece of text that's sent to the output will print on the same line. You can hover over each of these methods in your code to see their description.

7. Next, we'll use refactoring again to make the code a little more concise. Click on the variable `now` in the line `DateTime now = DateTime.Now;`.

Notice that a little screwdriver icon appears in the margin on that line.

8. Click the screwdriver icon to see what suggestions Visual Studio has available. In this case, it's showing the **Inline temporary variable** refactoring to remove a line of code without changing the overall behavior of the code:



9. Click **Inline temporary variable** to refactor the code.
10. Run the program again by pressing **Ctrl+F5**. The output looks something like this:

A screenshot of a Windows Command Prompt window titled 'cmd C:\WINDOWS\system32\cmd.exe'. The window contains the following text:

```
What is your name?  
Georgette  
  
Hello Georgette!  
Day of year: 151  
Press any key to continue . . .
```

10. Run the program again by pressing **Ctrl+F5**. The output looks something like this:

A screenshot of the Microsoft Visual Studio Debug Console window titled 'Microsoft Visual Studio Debug Console'. The window contains the following text:

```
What is your name?  
Georgette  
  
Hello Georgette!  
Day of year: 43  
  
C:\Program Files\dotnet\dotnet.exe (process 10744) exited with code 0.  
Press any key to close this window . . .
```

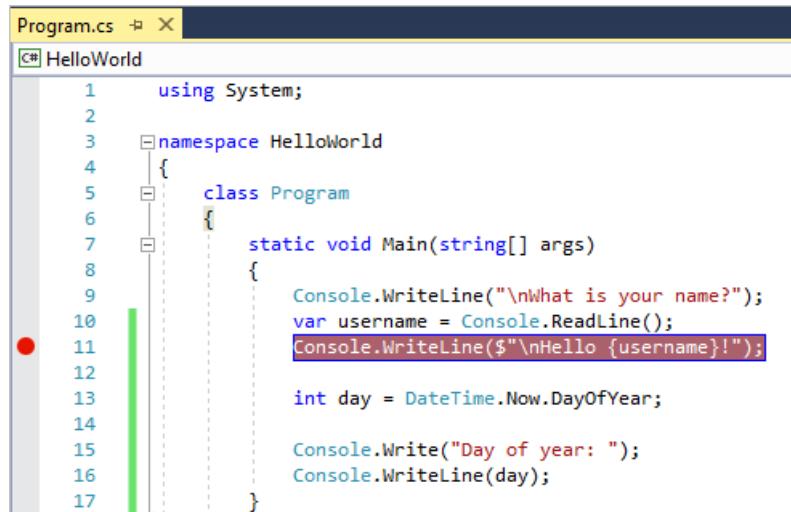
Debug code

When you write code, you need to run it and test it for bugs. Visual Studio's debugging system lets you step through code one statement at a time and inspect variables as you go. You can set *breakpoints* that stop execution of the code at a particular line. You can observe how the value of a variable changes as the code runs, and more.

Let's set a breakpoint to see the value of the `username` variable while the program is "in flight".

1. Find the line of code that says `Console.WriteLine($"\\nHello {username}!");`. To set a breakpoint on this line of code, that is, to make the program pause execution at this line, click in the far left margin of the editor. You can also click anywhere on the line of code and then press **F9**.

A red circle appears in the far left margin, and the code is highlighted in red.



The screenshot shows the Visual Studio code editor with a file named "Program.cs". The code is as follows:

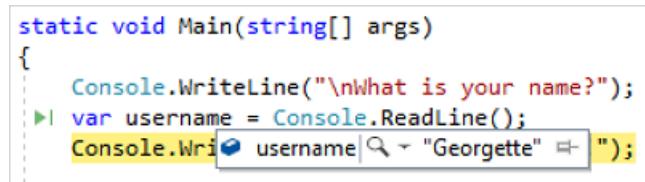
```
1  using System;
2
3  namespace HelloWorld
4  {
5      class Program
6      {
7          static void Main(string[] args)
8          {
9              Console.WriteLine("What is your name?");
10             var username = Console.ReadLine();
11             Console.WriteLine($"\\nHello {username}!");
12
13             int day = DateTime.Now.DayOfYear;
14
15             Console.Write("Day of year: ");
16             Console.WriteLine(day);
17         }
18     }
19 }
```

A red circle (breakpoint) is visible in the margin next to line 11. The line of code containing the breakpoint is highlighted in yellow.

2. Start debugging by selecting **Debug > Start Debugging** or by pressing **F5**.
3. When the console window appears and asks for your name, type it in and press **Enter**.

The focus returns to the Visual Studio code editor and the line of code with the breakpoint is highlighted in yellow. This signifies that it's the next line of code that the program will execute.

4. Hover your mouse over the `username` variable to see its value. Alternatively, you can right-click on `username` and select **Add Watch** to add the variable to the **Watch** window, where you can also see its value.



The screenshot shows the Visual Studio code editor with the same code as before. The line containing the breakpoint is highlighted in yellow. A tooltip is shown over the `username` variable, displaying the value "Georgette".

```
static void Main(string[] args)
{
    Console.WriteLine("What is your name?");
    var username = Console.ReadLine();
    Console.WriteLine($"\\nHello {username}!");
}
```

5. To let the program run to completion, press **F5** again.

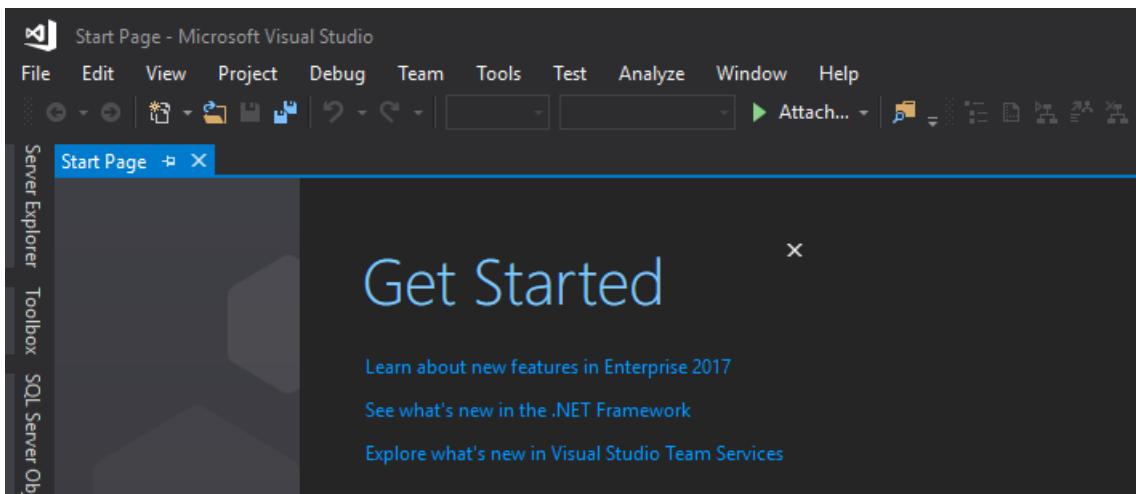
To get more details about debugging in Visual Studio, see [Debugger feature tour](#).

Customize Visual Studio

You can personalize the Visual Studio user interface, including change the default color theme. To change to the **Dark** theme:

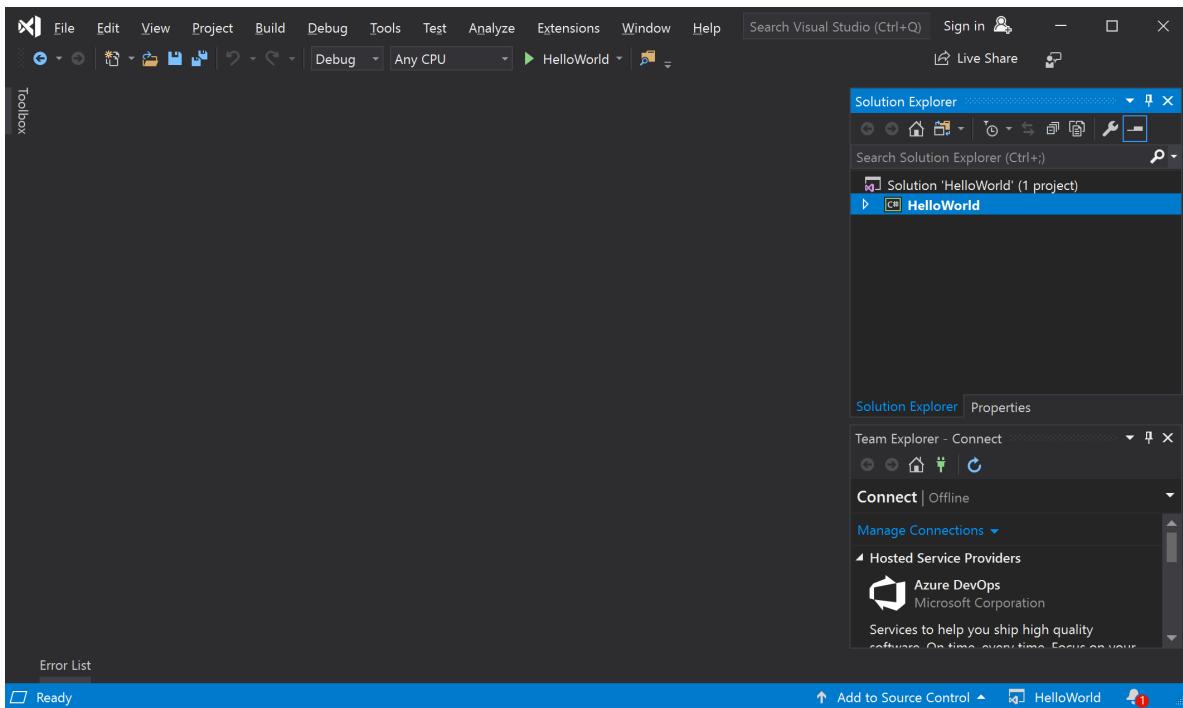
1. On the menu bar, choose **Tools > Options** to open the **Options** dialog.
2. On the **Environment > General** options page, change the **Color theme** selection to **Dark**, and then choose **OK**.

The color theme for the entire IDE changes to **Dark**.



2. On the **Environment > General** options page, change the **Color theme** selection to **Dark**, and then choose **OK**.

The color theme for the entire IDE changes to **Dark**.



To learn about other ways you can personalize the IDE, see [Personalize Visual Studio](#).

Next steps

Explore Visual Studio further by following along with one of these introductory articles:

- Get acquainted with the code editor in [Learn to use the code editor](#)
- Learn how Visual Studio organizes code in [Learn about projects and solutions](#)

If you're ready to dive into more coding, one of the following language-specific quickstarts is a good next step:

- [Use Visual Studio to create your first Python web app](#)
- [Use Visual Studio to create your first C# web app](#)
- [Use Visual Studio to create your first F# web app](#)

- Use Visual Studio to create your first Node.js app
- Use Visual Studio to create your first C++ console app

See also

- Discover [more Visual Studio features](#)
- Visit visualstudio.microsoft.com
- Read [The Visual Studio blog](#)

Learn to use the code editor

10/18/2019 • 5 minutes to read • [Edit Online](#)

In this 10-minute introduction to the code editor in Visual Studio, we'll add code to a file to look at some of the ways that Visual Studio makes writing, navigating, and understanding code easier.

TIP

If you haven't already installed Visual Studio, go to the [Visual Studio downloads](#) page to install it for free.

TIP

If you haven't already installed Visual Studio, go to the [Visual Studio downloads](#) page to install it for free.

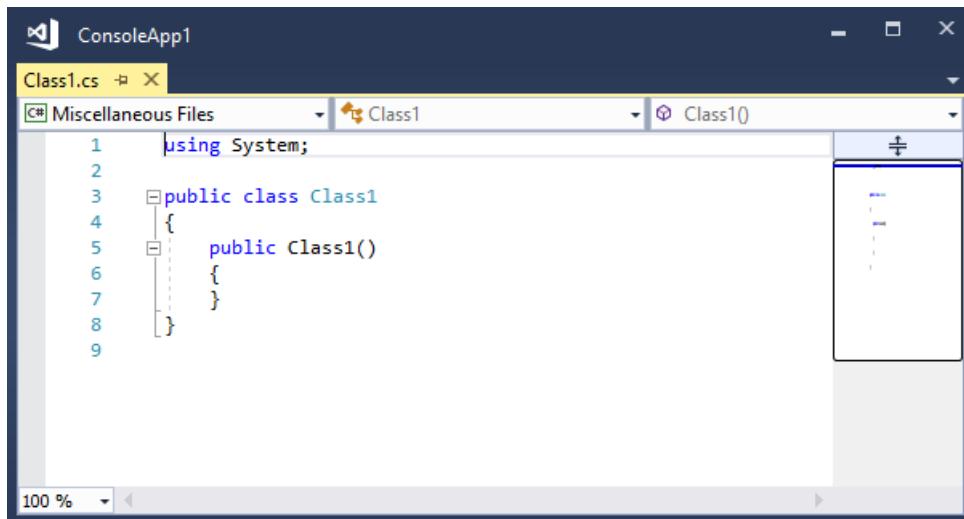
This article assumes you're already familiar with a programming language. If you aren't, we suggest you look at one of the programming quickstarts first, such as create a web app with [Python](#) or [C#](#), or create a console app with [Visual Basic](#) or [C++](#).

Create a new code file

Start by creating a new file and adding some code to it.

1. Open Visual Studio.
1. Open Visual Studio. Press **Esc** or click **Continue without code** on the start window to open the development environment.
2. From the **File** menu on the menu bar, choose **New > File**.
3. In the **New File** dialog box, under the **General** category, choose **Visual C# Class**, and then choose **Open**.

A new file opens in the editor with the skeleton of a C# class. (Notice that we don't have to create a full Visual Studio project to gain some of the benefits that the code editor offers; all you need is a code file!)

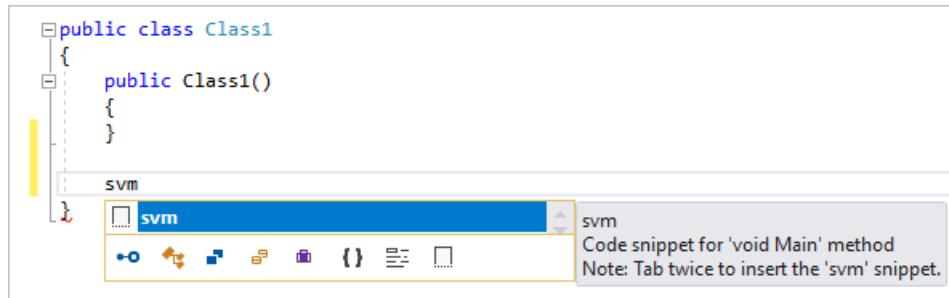


Use code snippets

Visual Studio provides useful *code snippets* that you can use to quickly and easily generate commonly used code blocks. [Code snippets](#) are available for different programming languages including C#, Visual Basic, and C+++. Let's add the C# `void Main` snippet to our file.

1. Place your cursor just above the final closing brace } in the file, and type the characters `svm`. (`svm` stands for `static void Main`; the `Main()` method is the entry point for C# applications.)

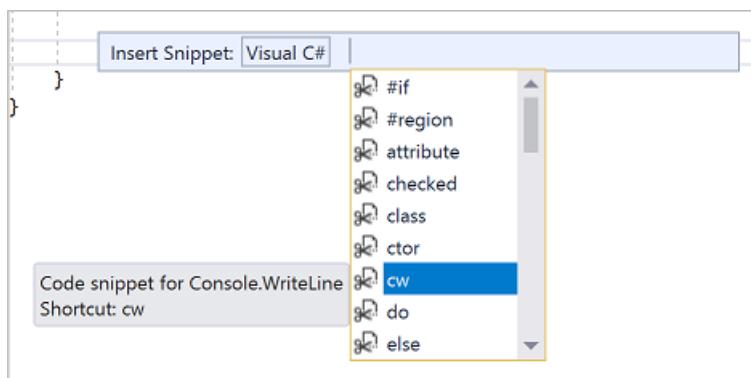
A pop-up dialog box appears with information about the `svm` code snippet.



2. Press **Tab** twice to insert the code snippet.

You see the `static void Main()` method signature get added to the file.

The available code snippets vary for different programming languages. You can look at the available code snippets for your language by choosing **Edit > IntelliSense > Insert Snippet**, and then choosing your language's folder. For C#, the list looks like this:



The list includes snippets for creating a `class`, a `constructor`, a `for` loop, an `if` or `switch` statement, and more.

Comment out code

The toolbar, which is the row of buttons under the menu bar in Visual Studio, can help make you more productive as you code. For example, you can toggle IntelliSense completion mode (IntelliSense is a coding aid that displays a list of matching methods, amongst other things), increase or decrease a line indent, or comment out code that you don't want to compile. In this section, we'll comment out some code.



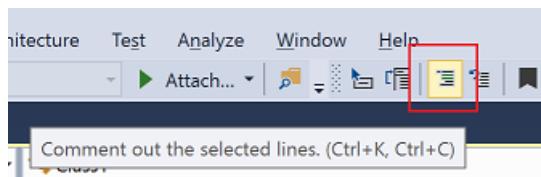
1. Paste the following code into the `Main()` method body.

```
// _words is a string array that we'll sort alphabetically
string[] _words = {
    "the",
    "quick",
    "brown",
    "fox",
    "jumps"
};

string[] morewords = {
    "over",
    "the",
    "lazy",
    "dog"
};

IEnumerable<string> query = from word in _words
                             orderby word.Length
                             select word;
```

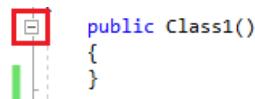
2. We're not using the `morewords` variable, but we may use it later so we don't want to completely delete it. Instead, let's comment out those lines. Select the entire definition of `morewords` to the closing semi-colon, and then choose the **Comment out the selected lines** button on the toolbar. If you prefer to use the keyboard, press **Ctrl+K, Ctrl+C**.



The C# comment characters `//` are added to the beginning of each selected line to comment out the code.

Collapse code blocks

We don't want to see the empty `constructor` for `Class1` that was generated, so to unclutter our view of the code, let's collapse it. Choose the small gray box with the minus sign inside it in the margin of the first line of the constructor. Or, if you're a keyboard user, place the cursor anywhere in the constructor code and press **Ctrl+M, Ctrl+M**.



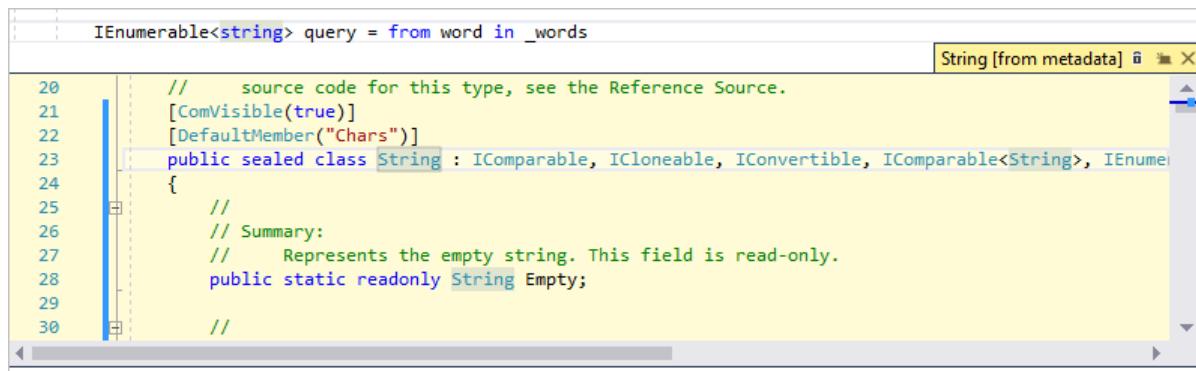
The code block collapses to just the first line, followed by an ellipsis (`...`). To expand the code block again, click the same gray box that now has a plus sign in it, or press **Ctrl+M, Ctrl+M** again. This feature is called **Outlining** and is especially useful when you're collapsing long methods or entire classes.

View symbol definitions

The Visual Studio editor makes it easy to inspect the definition of a type, method, etc. One way is to navigate to the file that contains the definition, for example by choosing **Go to Definition** anywhere the symbol is referenced. An even quicker way that doesn't move your focus away from the file you're working in is to use **Peek Definition**. Let's peek at the definition of the `string` type.

1. Right-click on any occurrence of `string` and choose **Peek Definition** from the content menu. Or, press **Alt+F12**.

A pop-up window appears with the definition of the `String` class. You can scroll within the pop-up window, or even peek at the definition of another type from the peeked code.



2. Close the peeked definition window by choosing the small box with an "x" at the top right of the pop-up window.

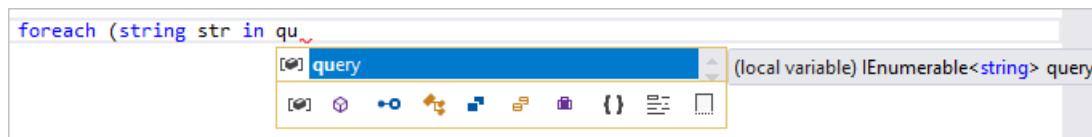
Use IntelliSense to complete words

IntelliSense is an invaluable resource when you're coding. It can show you information about available members of a type, or parameter details for different overloads of a method. You can also use IntelliSense to complete a word after you type enough characters to disambiguate it. Let's add a line of code to print out the ordered strings to the console window, which is the standard place for output from the program to go.

1. Below the `query` variable, start typing the following code:

```
foreach (string str in qu
```

You see IntelliSense show you **Quick Info** about the `query` symbol.



2. To insert the rest of the word `query` by using IntelliSense's word completion functionality, press **Tab**.
3. Finish off the code block to look like the following code. You can even practice using code snippets again by entering `cw` and then pressing **Tab** twice to generate the `Console.WriteLine` code.

```
foreach (string str in query)
{
    Console.WriteLine(str);
}
```

Refactor a name

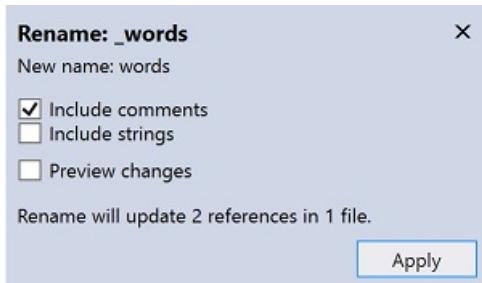
Nobody gets code right the first time, and one of the things you might have to change is the name of a variable or method. Let's try out Visual Studio's **refactor** functionality to rename the `_words` variable to `words`.

1. Place your cursor over the definition of the `_words` variable, and choose **Rename** from the right-click or context menu, or press **Ctrl+R, Ctrl+R**.

A pop-up **Rename** dialog box appears at the top right of the editor.

2. Enter the desired name `words`. Notice that the reference to `words` in the query is also automatically

renamed. Before you press **Enter**, select the **Include comments** checkbox in the **Rename** pop-up box.



3. Press **Enter**.

Both occurrences of `words` have been renamed, as well as the reference to `words` in the code comment.

Next steps

[Learn about projects and solutions](#)

See also

- [Code snippets](#)
- [Navigate code](#)
- [Outlining](#)
- [Go To Definition and Peek Definition](#)
- [Refactoring](#)
- [Use IntelliSense](#)

Learn about projects and solutions

10/18/2019 • 8 minutes to read • [Edit Online](#)

In this introductory article, we'll explore what it means to create a *solution* and a *project* in Visual Studio. A solution is a container that's used to organize one or more related code projects, for example a class library project and a corresponding test project. We'll look at the properties of a project and some of the files it can contain. We'll also create a reference from one project to another.

If you haven't already installed Visual Studio, go to the [Visual Studio downloads](#) page to install it for free.

If you haven't already installed Visual Studio, go to the [Visual Studio downloads](#) page to install it for free.

We'll construct a solution and project from scratch as an educational exercise to understand the concept of a project. In your general use of Visual Studio, you'll likely use some of the various project *templates* that Visual Studio offers when you create a new project.

NOTE

Solutions and projects aren't required to develop apps in Visual Studio. You can also just open a folder that contains code and start coding, building, and debugging. For example, if you clone a [GitHub](#) repo, it might not contain Visual Studio projects and solutions. For more information, see [Develop code in Visual Studio without projects or solutions](#).

Solutions and projects

Despite its name, a solution is not an "answer". A solution is simply a container used by Visual Studio to organize one or more related projects. When you open a solution in Visual Studio, it automatically loads all the projects that the solution contains.

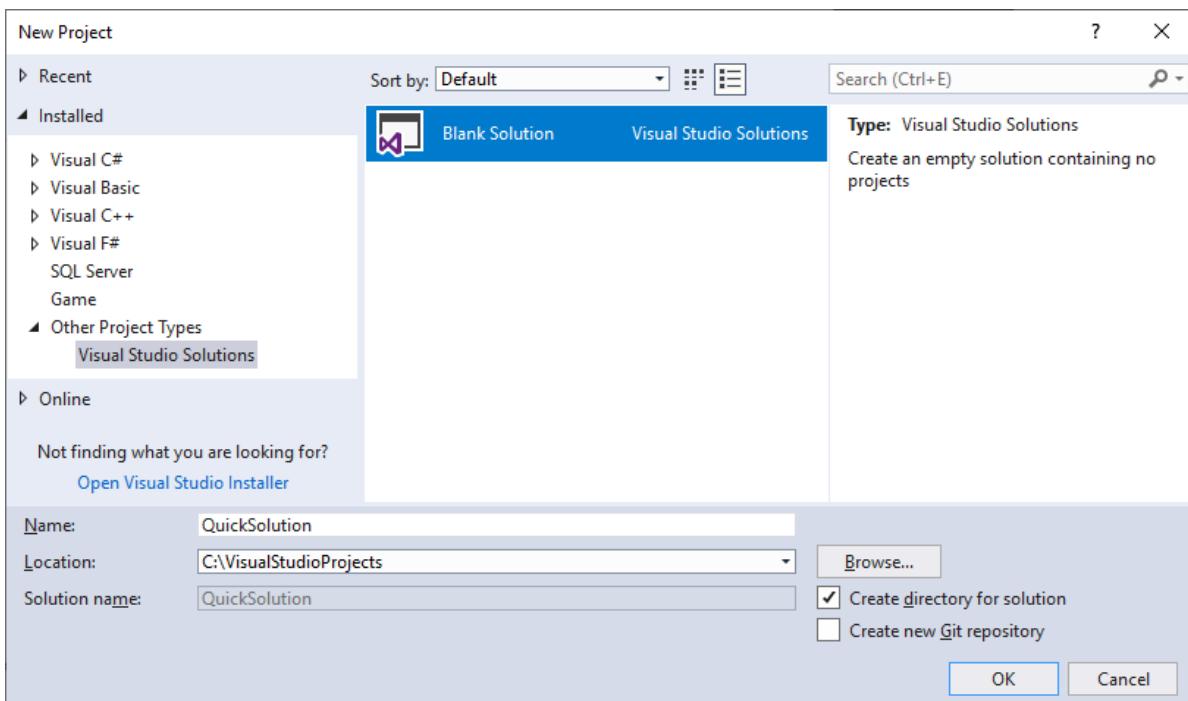
Create a solution

We'll start our exploration by creating an empty solution. After you get to know Visual Studio, you probably won't find yourself creating empty solutions very often. When you create a new project, Visual Studio automatically creates a solution to house the project if there's not a solution already open.

1. Open Visual Studio.
2. On the top menu bar, choose **File > New > Project**.

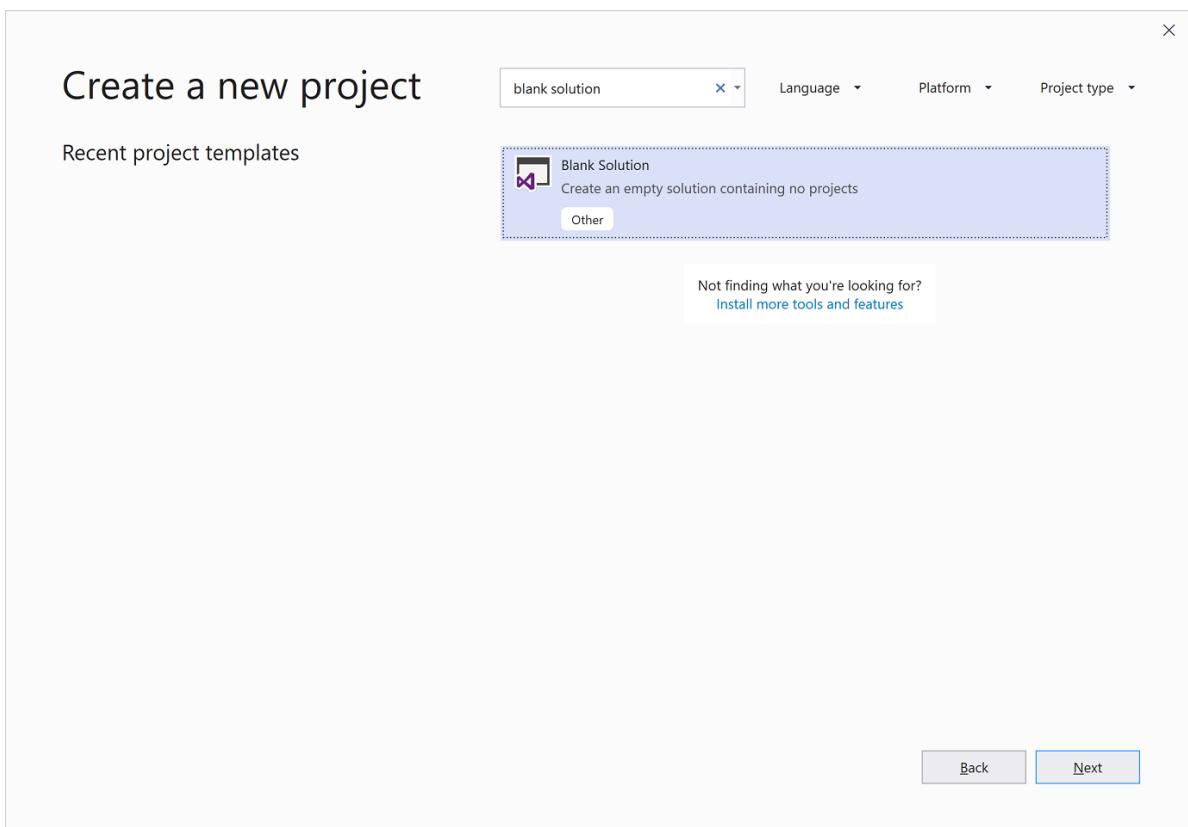
The **New Project** dialog box opens.

3. In the left pane, expand **Other Project Types**, then choose **Visual Studio Solutions**. In the center pane, choose the **Blank Solution** template. Name your solution **QuickSolution**, then choose the **OK** button.



The **Start Page** closes, and a solution appears in **Solution Explorer** on the right-hand side of the Visual Studio window. You'll probably use **Solution Explorer** often, to browse the contents of your projects.

1. Open Visual Studio.
2. On the start window, choose **Create a new project**.
3. On the **Create a new project** page, enter **blank solution** into the search box, select the **Blank Solution** template, and then choose **Next**.



4. Name the solution **QuickSolution**, and then choose **Create**.

A solution appears in **Solution Explorer** on the right-hand side of the Visual Studio window. You'll probably use **Solution Explorer** often, to browse the contents of your projects.

Add a project

Now let's add our first project to the solution. We'll start with an empty project and add the items we need to the project.

- From the right-click or context menu of **Solution 'QuickSolution'** in **Solution Explorer**, choose **Add > New Project**.

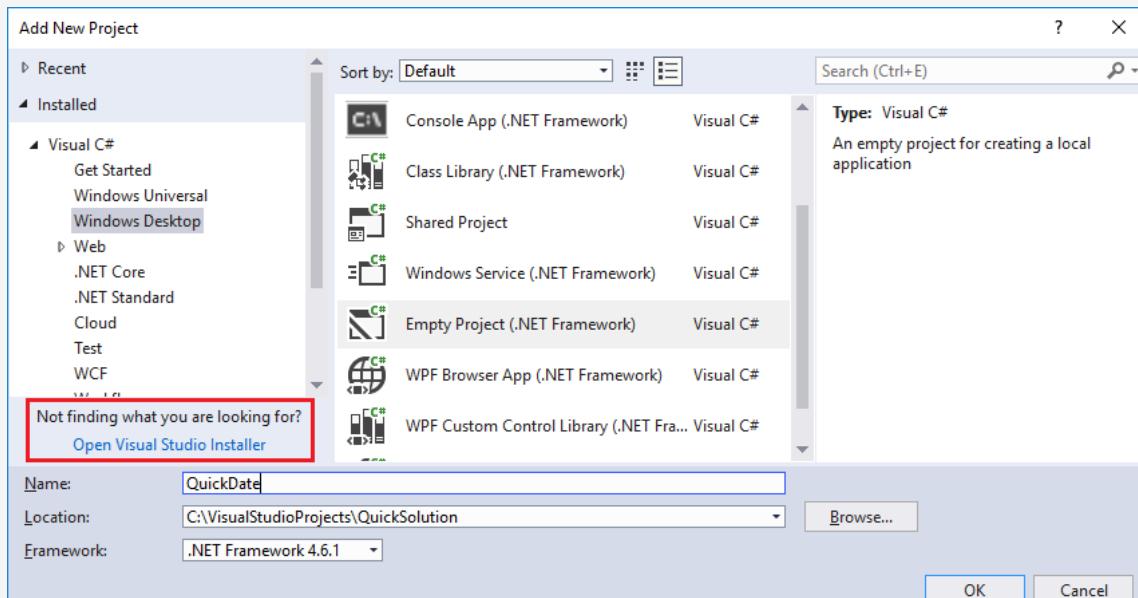
The **Add New Project** dialog box opens.

- In the left pane, expand **Visual C#** and choose **Windows Desktop**. Then, in the middle pane, choose the **Empty Project (.NET Framework)** template. Name the project **QuickDate**, then choose **OK**.

A project named QuickDate appears beneath the solution in **Solution Explorer**. Currently it contains a single file called *App.config*.

NOTE

If you don't see **Visual C#** in the left pane of the dialog box, you need to install the **.NET desktop development** Visual Studio workload. Visual Studio uses workload-based installation to only install the components you need for the type of development you do. An easy way to install a new workload is to choose the **Open Visual Studio Installer** link in the bottom left corner of the **Add New Project** dialog box. After Visual Studio Installer launches, choose the **.NET desktop development** workload and then the **Modify** button.



- From the right-click or context menu of **Solution 'QuickSolution'** in **Solution Explorer**, choose **Add > New Project**.

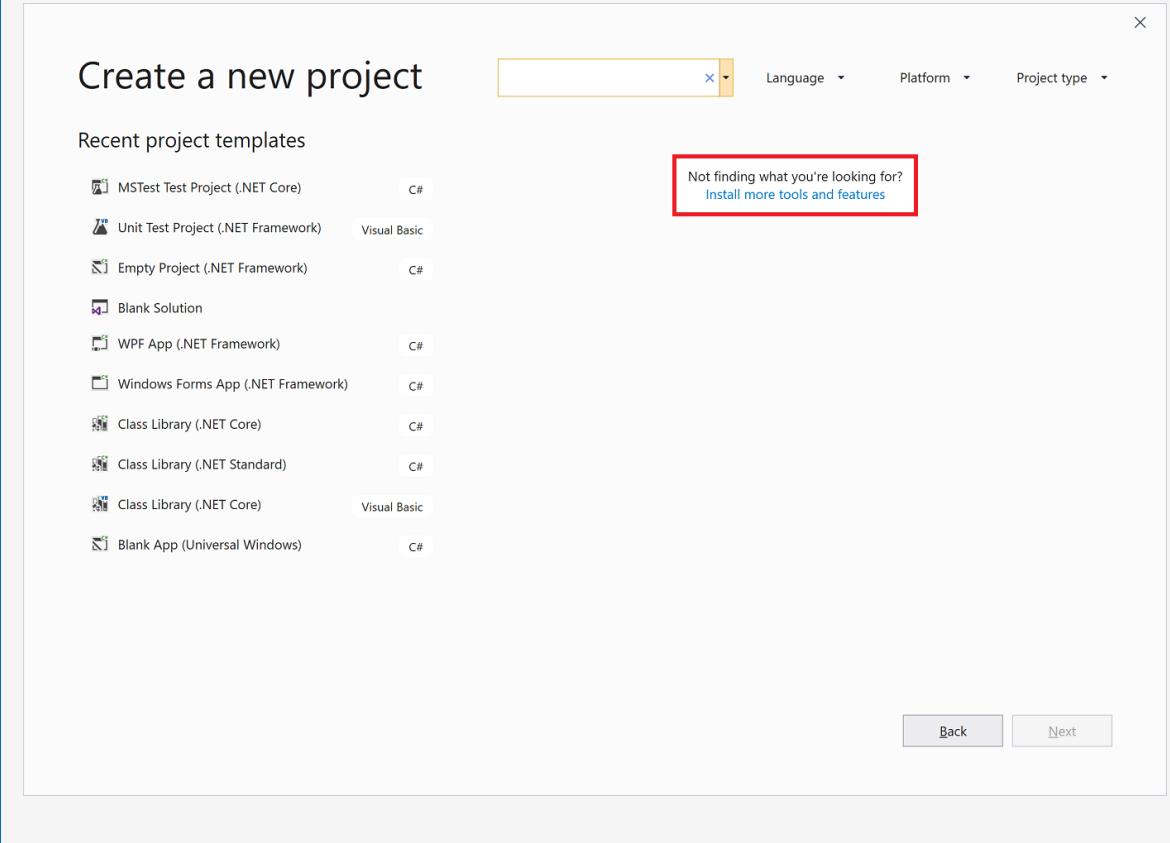
A dialog box opens that says **Add a new project**.

- Enter the text **empty** into the search box at the top, and then select **C#** under **Language**.
- Select the **Empty Project (.NET Framework)** template, and then choose **Next**.
- Name the project **QuickDate**, then choose **Create**.

A project named QuickDate appears beneath the solution in **Solution Explorer**. Currently it contains a single file called *App.config*.

NOTE

If you don't see the **Empty Project (.NET Framework)** template, you need to install the **.NET desktop development** Visual Studio *workload*. Visual Studio uses workload-based installation to only install the components you need for the type of development you do. An easy way to install a new workload when you're creating a new project is to choose the **Install more tools and features** link under the text that says **Not finding what you're looking for?**. After Visual Studio Installer launches, choose the **.NET desktop development** workload and then the **Modify** button.



Add an item to the project

We have an empty project. Let's add a code file.

1. From the right-click or context menu of the **QuickDate** project in **Solution Explorer**, choose **Add > New Item**.

The **Add New Item** dialog box opens.

2. Expand **Visual C# Items**, then choose **Code**. In the middle pane choose the **Class** item template. Name the class **Calendar**, and then choose the **Add** button.

A file named *Calendar.cs* is added to the project. The *.cs* on the end is the file extension that is given to C# code files. The file appears in the visual project hierarchy in **Solution Explorer**, and its contents are opened in the editor.

3. Replace the contents of the *Calendar.cs* file with the following code:

```

using System;

namespace QuickDate
{
    internal class Calendar
    {
        static void Main(string[] args)
        {
            DateTime now = GetCurrentDate();
            Console.WriteLine($"Today's date is {now}");
            Console.ReadLine();
        }

        internal static DateTime GetCurrentDate()
        {
            return DateTime.Now.Date;
        }
    }
}

```

You don't need to understand what the code does, but if you want, you can run the program by pressing **Ctrl+F5** and see that it prints today's date to the console (or standard output) window.

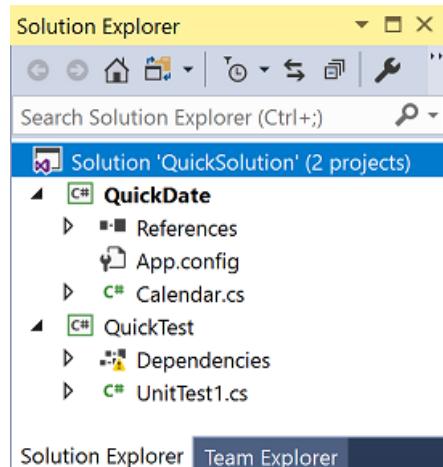
Add a second project

It is common for solutions to contain more than one project, and often these projects reference each other. Some projects in a solution might be class libraries, some executable applications, and some might be unit test projects or websites.

Let's add a unit test project to our solution. This time we'll start from a project template so we don't have to add an additional code file to the project.

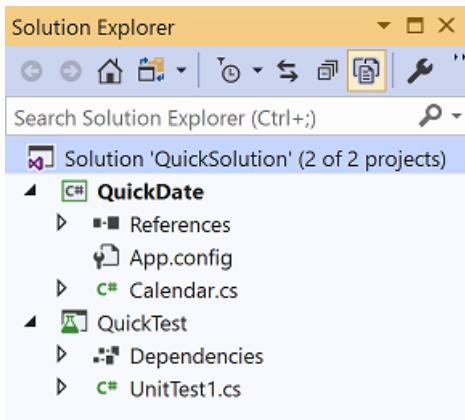
1. From the right-click or context menu of **Solution 'QuickSolution'** in **Solution Explorer**, choose **Add > New Project**.
2. In the left pane, expand **Visual C#** and choose the **Test** category. In the middle pane, choose the **MSTest Test Project (.NET Core)** project template. Name the project **QuickTest**, and then choose **OK**.

A second project is added to **Solution Explorer**, and a file named *UnitTest1.cs* opens in the editor.



2. In the **Add a new project** dialog box, enter the text **unit test** into the search box at the top, and then select **C#** under **Language**.
3. Choose the **MSTest Test Project (.NET Core)** project template, and then choose **Next**.
4. Name the project **QuickTest**, and then choose **Create**.

A second project is added to **Solution Explorer**, and a file named *UnitTest1.cs* opens in the editor.



Add a project reference

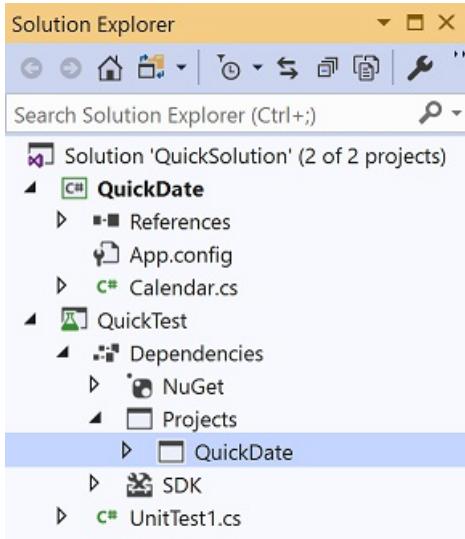
We're going to use the new unit test project to test our method in the **QuickDate** project, so we need to add a reference to that project. This creates a *build dependency* between the two projects, meaning that when you build the solution, **QuickDate** is built before **QuickTest**.

1. Choose the **Dependencies** node in the **QuickTest** project, and from the right-click or context menu, choose **Add Reference**.

The **Reference Manager** dialog box opens.

2. In the left pane, expand **Projects** and choose **Solution**. In the middle pane, choose the checkbox next to **QuickDate**, and then choose **OK.

A reference to the **QuickDate** project is added.



Add test code

1. Now we'll add test code to the C# test code file. Replace the contents of *UnitTest1.cs* with the following code:

```
using System;
using Microsoft.VisualStudio.TestTools.UnitTesting;

namespace QuickTest
{
    [TestClass]
    public class UnitTest1
    {
        [TestMethod]
        public void TestGetCurrentDate()
        {
            Assert.AreEqual(DateTime.Now.Date, QuickDate.Calendar.GetCurrentDate());
        }
    }
}
```

You'll see a red squiggle under some of the code. We'll fix this error by making the test project a [friend assembly](#) to the **QuickDate** project.

2. Back in the **QuickDate** project, open the *Calendar.cs* file if it's not already open. Add the following [using statement](#) and [InternalsVisibleToAttribute](#) attribute to the top of the file to resolve the error in the test project.

```
using System.Runtime.CompilerServices;

[assembly: InternalsVisibleTo("QuickTest")]
```

The code file should look like this:

```
1  using System;
2  using System.Runtime.CompilerServices;
3
4  [assembly: InternalsVisibleTo("QuickTest")]
5
6  namespace QuickDate
7  {
8      internal class Calendar
9      {
10          static void Main(string[] args)
11          {
12              DateTime now = GetCurrentDate();
13              Console.WriteLine($"Today's date is {now}");
14              Console.ReadLine();
15          }
16
17          internal static DateTime GetCurrentDate()
18          {
19              return DateTime.Now.Date;
20          }
21      }
22  }
```

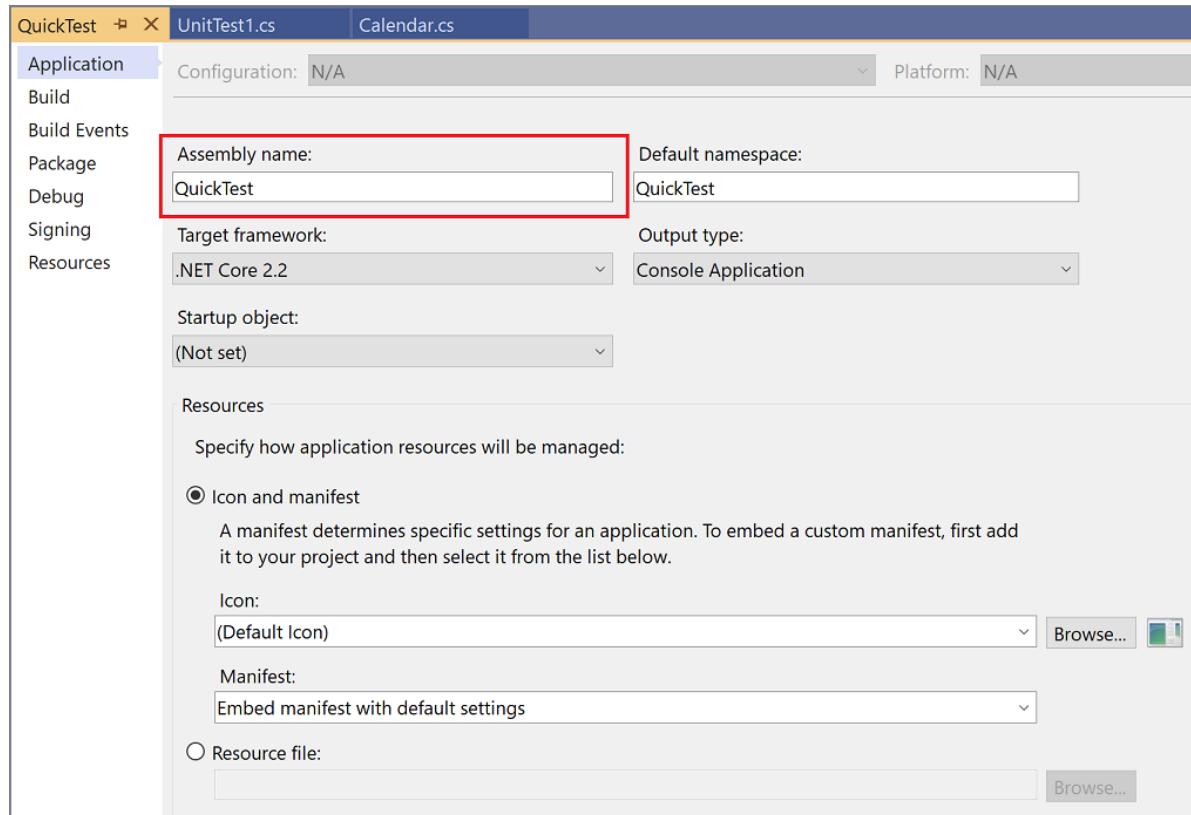
Project properties

The line in the *Calendar.cs* file that contains the [InternalsVisibleToAttribute](#) attribute references the assembly name (file name) of the **QuickTest** project. The assembly name might not always be the same as the project name. To find the assembly name of a project, open the project properties.

1. In **Solution Explorer**, select the **QuickTest** project. From the right-click or context menu, select

Properties, or just press **Alt+Enter**.

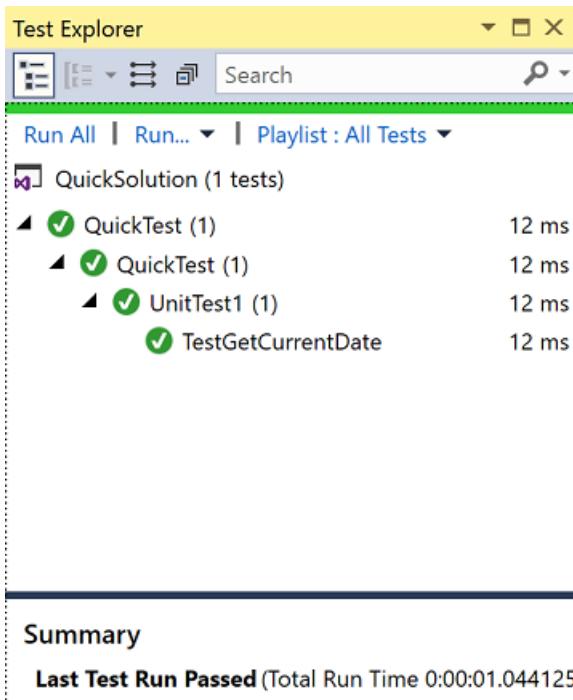
The *property pages* for the project open on the **Application** tab. The property pages contain various settings for the project. Notice that the assembly name of the **QuickTest** project is indeed "QuickTest". If you wanted to change it, this is where you'd do that. Then, when you build the test project, the name of the resulting binary file would change from *QuickTest.dll* to whatever you chose.



2. Explore some of the other tabs of the project's property pages, such as **Build** and **Debug**. These tabs are different for different types of projects.

Next steps

If you want to check that your unit test is working, choose **Test > Run > All Tests** from the menu bar. A window called **Test Explorer** opens, and you should see that the **TestGetCurrentDate** test passes.



TIP

If **Test Explorer** doesn't open automatically, open it by choosing **Test > Windows > Test Explorer** from the menu bar.

See also

- [Create projects and solutions](#)
- [Manage project and solution properties](#)
- [Manage references in a project](#)
- [Develop code in Visual Studio without projects or solutions](#)
- [Visual Studio IDE overview](#)

Features of Visual Studio

10/25/2019 • 7 minutes to read • [Edit Online](#)

The [Visual Studio IDE overview](#) article gives a basic introduction to Visual Studio. This article describes features that might be more appropriate for experienced developers, or those developers who are already familiar with Visual Studio.

Modular installation

Visual Studio's modular installer enables you to choose and install *workloads*. Workloads are groups of features needed for the programming language or platform you prefer. This strategy helps to keep the footprint of the Visual Studio installation smaller, which means it installs and updates faster too.

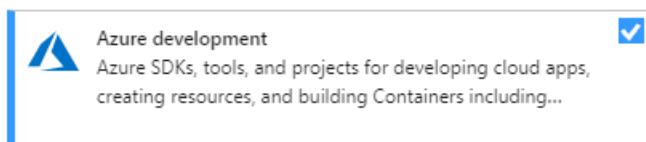
If you haven't already installed Visual Studio, go to the [Visual Studio downloads](#) page to install it for free.

If you haven't already installed Visual Studio, go to the [Visual Studio downloads](#) page to install it for free.

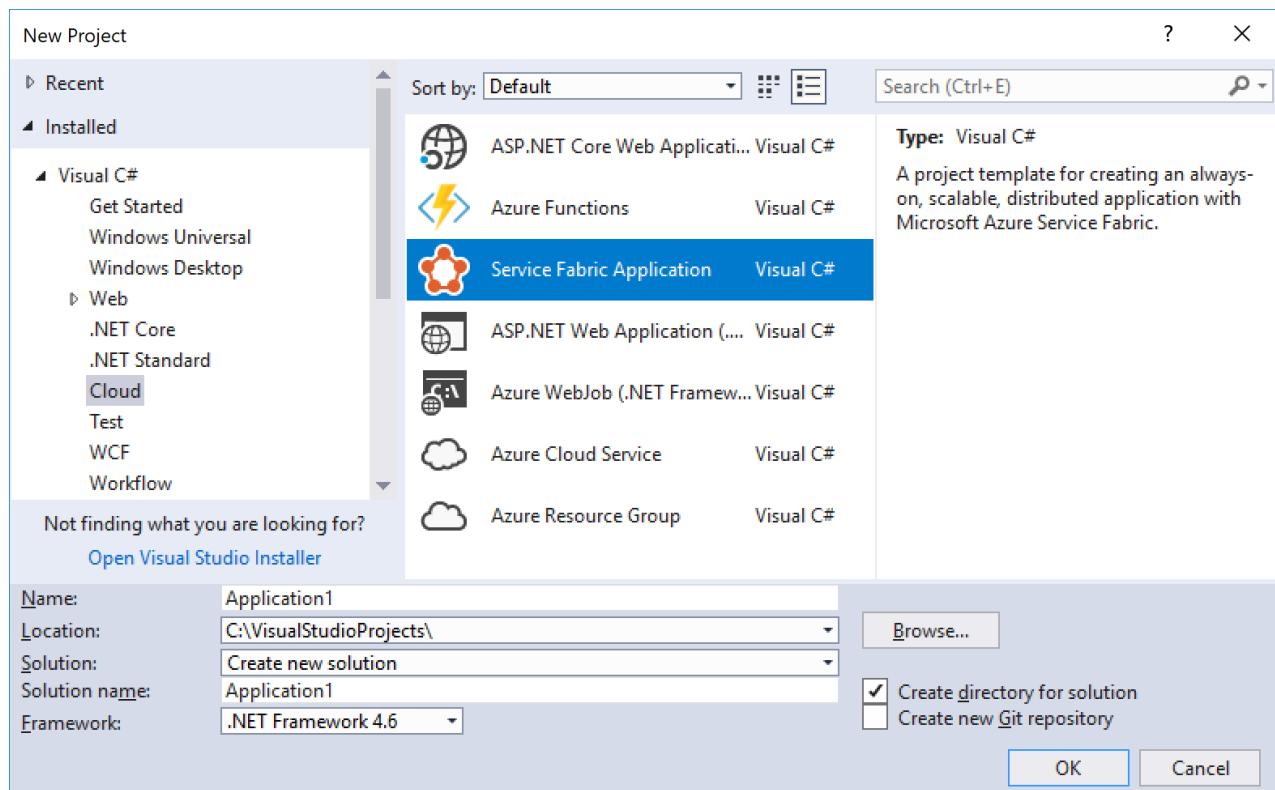
To learn more about setting up Visual Studio on your system, see [Install Visual Studio](#).

Create cloud-enabled apps for Azure

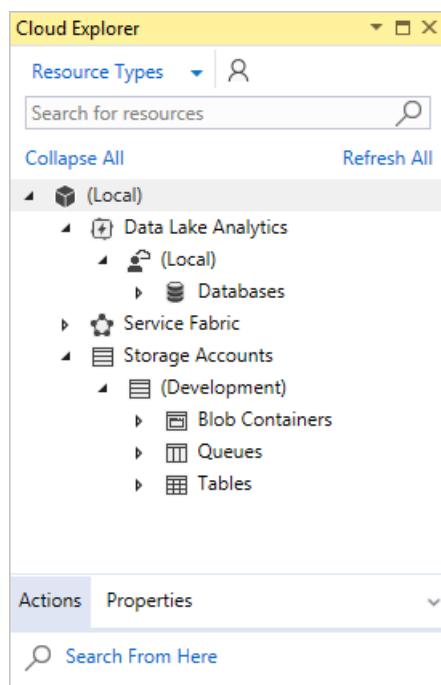
Visual Studio offers a suite of tools that enable you to easily create cloud-enabled applications powered by Microsoft Azure. You can configure, build, debug, package, and deploy applications and services on Microsoft Azure directly from the IDE. To get the Azure tools and project templates, select the **Azure development** workload when you install Visual Studio.



After you install the **Azure development** workload, the following **Cloud** templates for C# are available in the **New Project** dialog:



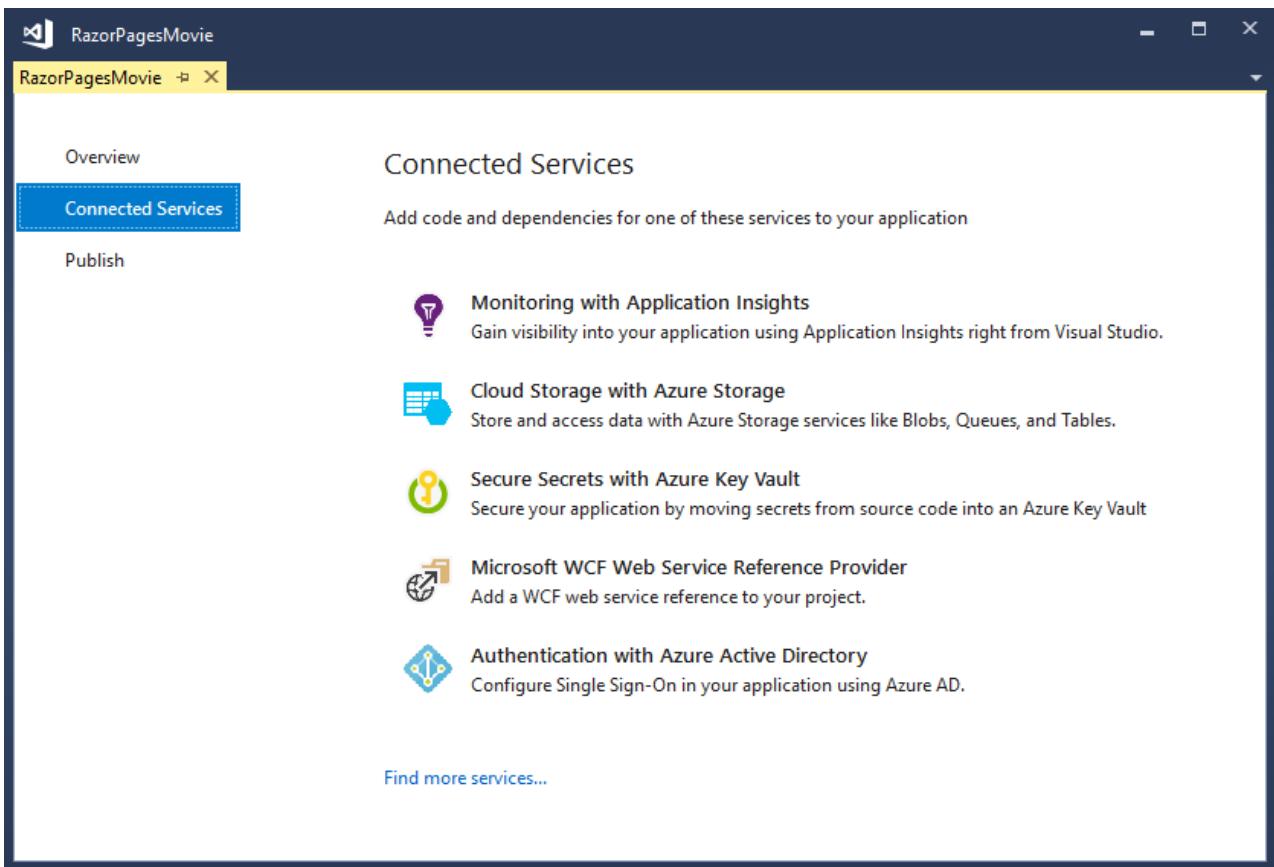
Visual Studio's **Cloud Explorer** lets you view and manage your Azure-based cloud resources within Visual Studio. These resources may include virtual machines, tables, SQL databases, and more. **Cloud Explorer** shows the Azure resources in all the accounts managed under the Azure subscription you're logged into. And if a particular operation requires the Azure portal, **Cloud Explorer** provides links that take you to the place in the portal where you need to go.



You can leverage Azure services for your apps using **Connected Services** such as:

- [Active Directory connected service](#) so users can use their accounts from [Azure Active Directory](#) to connect to web apps
- [Azure Storage connected service](#) for blob storage, queues, and tables
- [Key Vault connected service](#) to manage secrets for web apps

The available **Connected Services** depend on your project type. Add a service by right-clicking on the project in **Solution Explorer** and choosing **Add > Connected Service**.



For more information, see [Move to the cloud With Visual Studio and Azure](#).

Create apps for the web

The web drives our modern world, and Visual Studio can help you write apps for it. You can create web apps using ASP.NET, Node.js, Python, JavaScript, and TypeScript. Visual Studio understands web frameworks like Angular, jQuery, Express, and more. ASP.NET Core and .NET Core run on Windows, Mac, and Linux operating systems. [ASP.NET Core](#) is a major update to MVC, WebAPI and SignalR, and runs on Windows, Mac, and Linux. ASP.NET Core has been designed from the ground up to provide you with a lean and composable .NET stack for building modern cloud-based web apps and services.

For more information, see [Modern web tooling](#).

Build cross-platform apps and games

You can use Visual Studio to build apps and games for macOS, Linux, and Windows, as well as for Android, iOS, and other [mobile devices](#).

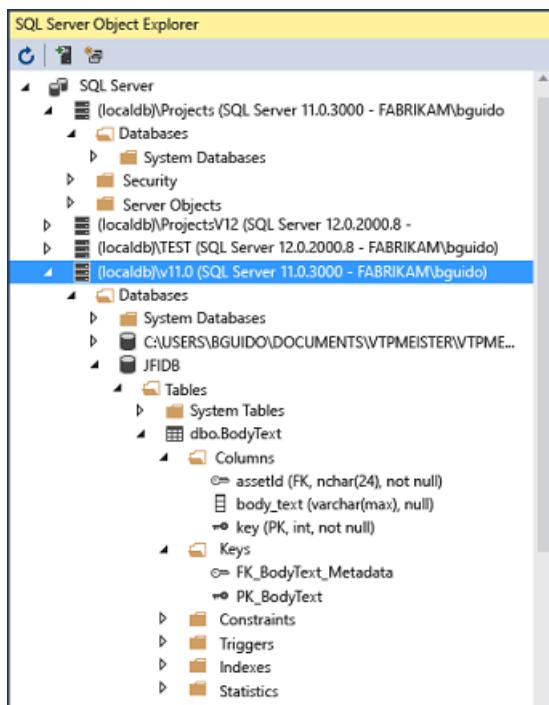
- Build [.NET Core](#) apps that run on Windows, macOS, and Linux.
- Build mobile apps for iOS, Android, and Windows in C# and F# by using [Xamarin](#).
- Use standard web technologies—HTML, CSS, and JavaScript—to build mobile apps for iOS, Android, and Windows by using [Apache Cordova](#).
- Build 2D and 3D games in C# by using [Visual Studio Tools for Unity](#).
- Build native C++ apps for iOS, Android, and Windows devices. Share common code in libraries built for iOS, Android, and Windows, by using [C++ for cross-platform development](#).
- Deploy, test, and debug Android apps with the [Android emulator](#).

Connect to databases

Server Explorer helps you browse and manage SQL Server instances and assets locally, remotely, and on Azure, Salesforce.com, Office 365, and websites. To open **Server Explorer**, on the main menu, choose **View > Server Explorer**. For more information on using Server Explorer, see [Add new connections](#).

SQL Server Data Tools (SSDT) is a powerful development environment for SQL Server, Azure SQL Database, and Azure SQL Data Warehouse. It enables you to build, debug, maintain, and refactor databases. You can work with a database project, or directly with a connected database instance on- or off-premises.

SQL Server Object Explorer in Visual Studio provides a view of your database objects similar to SQL Server Management Studio. SQL Server Object Explorer enables you to do light-duty database administration and design work. Work examples include editing table data, comparing schemas, executing queries by using contextual menus right from SQL Server Object Explorer, and more.



Debug, test, and improve your code

When you write code, you need to run it and test it for bugs and performance. Visual Studio's cutting-edge debugging system enables you to debug code running in your local project, on a remote device, or on a [device emulator](#). You can step through code one statement at a time and inspect variables as you go. You can set breakpoints that are only hit when a specified condition is true. Debug options can be managed in the code editor itself, so that you don't have to leave your code. To get more details about debugging in Visual Studio, see [First look at the debugger](#).

To learn more about improving the performance of your apps, checkout out Visual Studio's [profiling](#) feature.

For [testing](#), Visual Studio offers unit testing, Live Unit Testing, IntelliTest, load and performance testing, and more. Visual Studio also has advanced [code analysis](#) capabilities to catch design, security, and other types of flaws.

Deploy your finished application

When your application is ready to deploy to users or customers, Visual Studio provides the tools to do that. Deployment options include to Microsoft Store, to a SharePoint site, or with InstallShield or Windows Installer technologies. It's all accessible through the IDE. For more information, see [Deploy applications, services, and components](#).

Manage your source code and collaborate with others

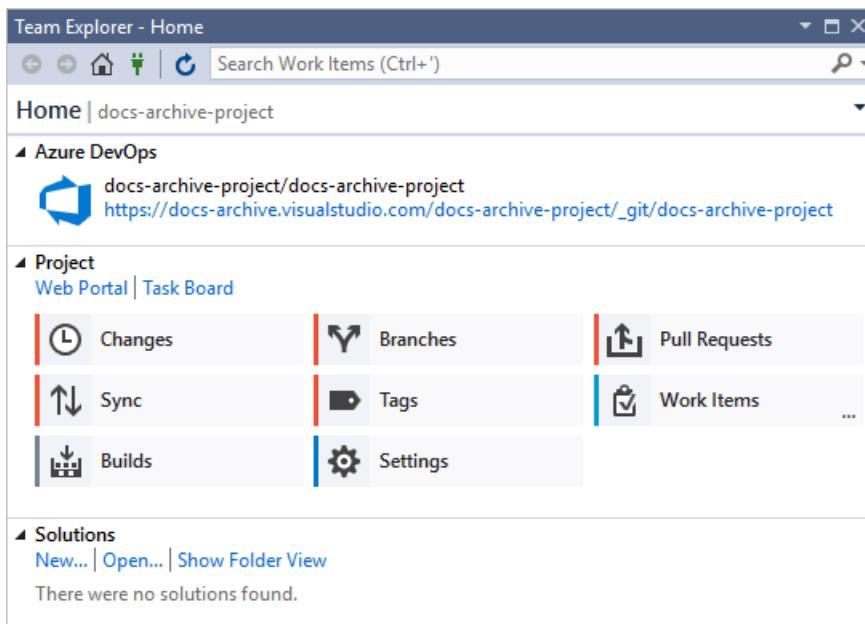
You can manage your source code in Git repos hosted by any provider, including GitHub. Or use [Azure DevOps Services](#) to manage code alongside bugs and work items for your whole project. See [Get started with Git and Azure Repos](#) to learn more about managing Git repos in Visual Studio using Team Explorer. Visual Studio also has other built-in source control features. To learn more about them, see [New Git features in Visual Studio \(blog\)](#).

Azure DevOps Services are cloud-based services to plan, host, automate, and deploy software and enable collaboration in teams. Azure DevOps Services support both Git repos (distributed version control) and Team Foundation Version Control (centralized version control). They support pipelines for continuous build and release (CI/CD) of code stored in version control systems. Azure DevOps Services also support Scrum, CMMI and Agile development methodologies.

Team Foundation Server (TFS) is the application lifecycle management hub for Visual Studio. It enables everyone involved with the development process to participate using a single solution. TFS is useful for managing heterogeneous teams and projects, too.

If you have an Azure DevOps organization or a Team Foundation Server on your network, you connect to it through the **Team Explorer** window in Visual Studio. From this window you can check code into or out of source control, manage work items, start builds, and access team rooms and workspaces. You can open **Team Explorer** from the search box, or on the main menu from **View > Team Explorer** or from **Team > Manage Connections**.

The following image shows the **Team Explorer** window for a solution that is hosted in Azure DevOps Services.



You can also automate your build process to build the code that the devs on your team have checked into version control. For example, you can build one or more projects nightly or every time that code is checked in. For more information, see [Azure Pipelines](#).

Extend Visual Studio

If Visual Studio doesn't have the exact functionality you need, you can add it! You can personalize the IDE based on your workflow and style, add support for external tools not yet integrated with Visual Studio, and modify existing functionality to increase your productivity. To find the latest version of the Visual Studio Extensibility Tools (VS SDK), see [Visual Studio SDK](#).

You can use the .NET Compiler Platform ("Roslyn") to write your own code analyzers and code generators. Find everything you need at [Roslyn](#).

Find [existing extensions](#) for Visual Studio created by Microsoft developers as well as our development community.

To learn more about extending Visual Studio, see [Extend Visual Studio IDE](#).

See also

- [Visual Studio IDE overview](#)
- [What's new in Visual Studio 2017](#)
- [What's new in Visual Studio 2019](#)

Install Visual Studio

10/7/2019 • 7 minutes to read • [Edit Online](#)

Welcome to Visual Studio 2019! In this version, it's easy to choose and install just the features you need. And because of its reduced minimum footprint, it installs quickly and with less system impact.

Welcome to a new way to install Visual Studio! In this version, we've made it easier for you to choose and install just the features you need. We've also reduced the minimum footprint of Visual Studio so that it installs more quickly and with less system impact than ever before.

NOTE

This topic applies to Visual Studio on Windows. For Visual Studio for Mac, see [Install Visual Studio for Mac](#).

Want to know more about what else is new in this version? See our [release notes](#).

Want to know more about what else is new in this version? See our [release notes](#).

Ready to install? We'll walk you through it, step-by-step.

Step 1 - Make sure your computer is ready for Visual Studio

Before you begin installing Visual Studio:

1. Check the [system requirements](#). These requirements help you know whether your computer supports Visual Studio 2017.
2. Apply the latest Windows updates. These updates ensure that your computer has both the latest security updates and the required system components for Visual Studio.
3. Reboot. The reboot ensures that any pending installs or updates don't hinder the Visual Studio install.
4. Free up space. Remove unneeded files and applications from your %SystemDrive% by, for example, running the Disk Cleanup app.
1. Check the [system requirements](#). These requirements help you know whether your computer supports Visual Studio 2019.
2. Apply the latest Windows updates. These updates ensure that your computer has both the latest security updates and the required system components for Visual Studio.
3. Reboot. The reboot ensures that any pending installs or updates don't hinder the Visual Studio install.
4. Free up space. Remove unneeded files and applications from your %SystemDrive% by, for example, running the Disk Cleanup app.

For questions about running previous versions of Visual Studio side by side with Visual Studio 2017, see the [Visual Studio compatibility details](#).

For questions about running previous versions of Visual Studio side by side with Visual Studio 2019, see the [Visual Studio 2019 Platform Targeting and Compatibility](#) page.

Step 2 - Download Visual Studio

Next, download the Visual Studio bootstrapper file.

To get a bootstrapper for Visual Studio 2017, see the [Visual Studio previous versions](#) download page for details on how to do so.

To do so, choose the following button, choose the edition of Visual Studio that you want, choose **Save**, and then choose **Open folder**.



Step 3 - Install the Visual Studio installer

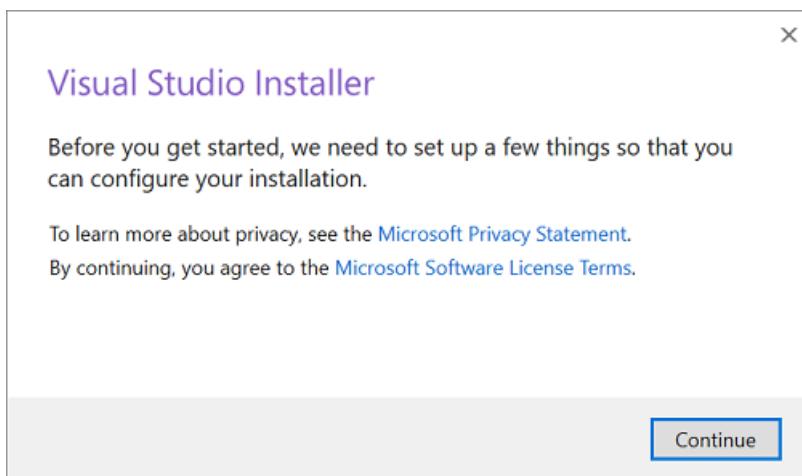
Run the bootstrapper file to install the Visual Studio Installer. This new lightweight installer includes everything you need to both install and customize Visual Studio.

1. From your **Downloads** folder, double-click the bootstrapper that matches or is similar to one of the following files:

- **vs_community.exe** for Visual Studio Community
- **vs_professional.exe** for Visual Studio Professional
- **vs_enterprise.exe** for Visual Studio Enterprise

If you receive a User Account Control notice, choose **Yes**.

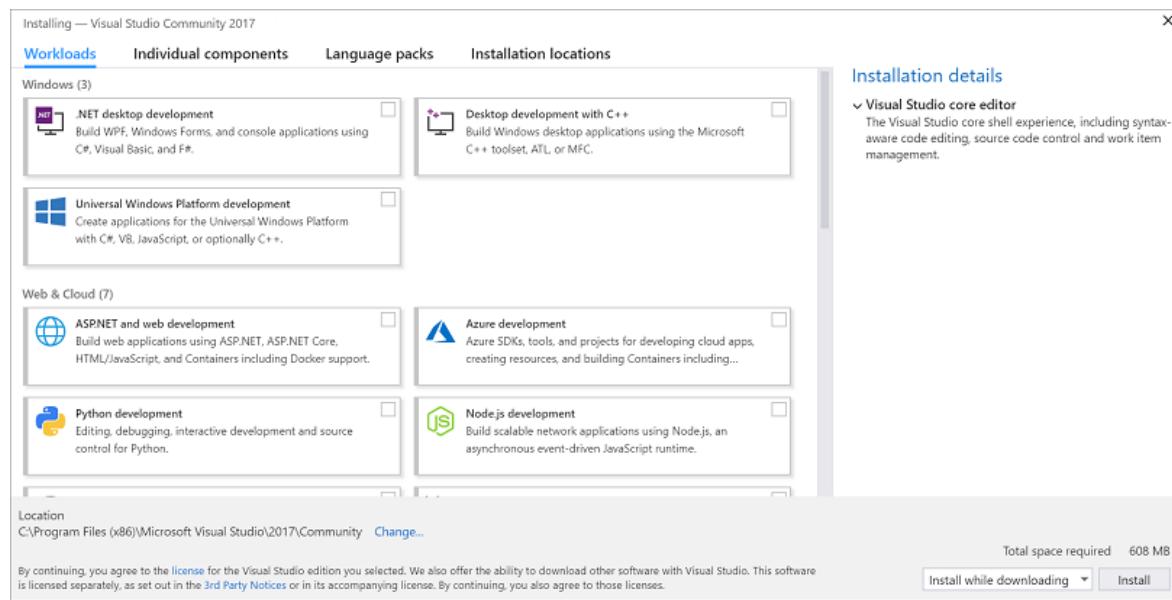
2. We'll ask you to acknowledge the Microsoft [License Terms](#) and the Microsoft [Privacy Statement](#). Choose **Continue**.



Step 4 - Choose workloads

After the installer is installed, you can use it to customize your installation by selecting the feature sets—or workloads—that you want. Here's how.

1. Find the workload you want in the **Installing Visual Studio** screen.

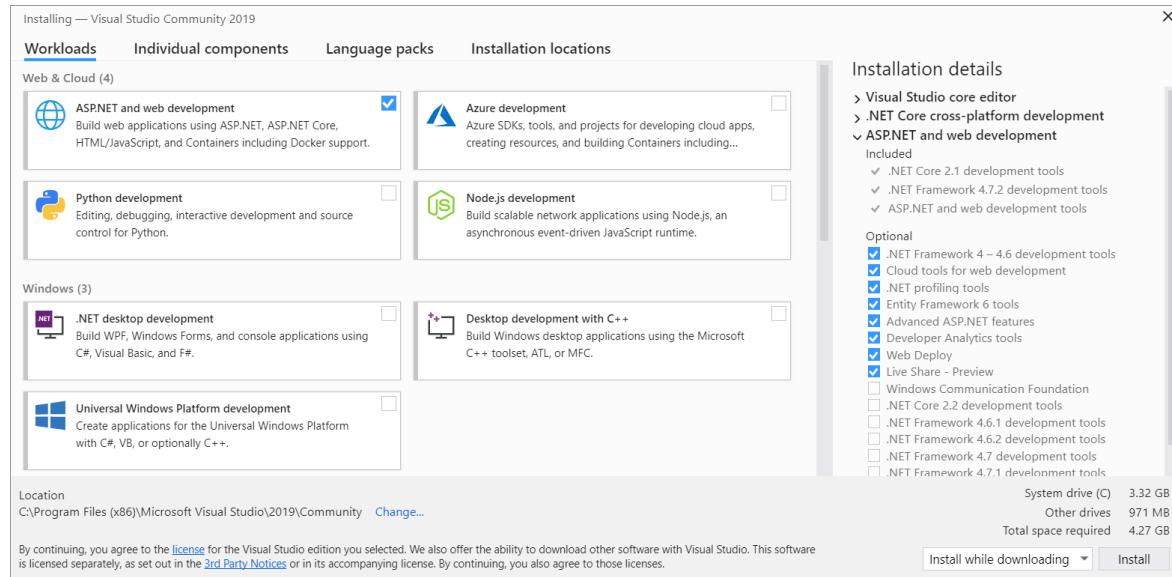


For example, choose the ".NET desktop development" workload. It comes with the default core editor, which includes basic code editing support for over 20 languages, the ability to open and edit code from any folder without requiring a project, and integrated source code control.

2. After you choose the workload(s) you want, choose **Install**.

Next, status screens appear that show the progress of your Visual Studio installation.

1. After the new workloads and components are installed, choose **Launch**.



For example, choose the "ASP.NET and web development" workload. It comes with the default core editor, which includes basic code editing support for over 20 languages, the ability to open and edit code from any folder without requiring a project, and integrated source code control.

2. After you choose the workload(s) you want, choose **Install**.

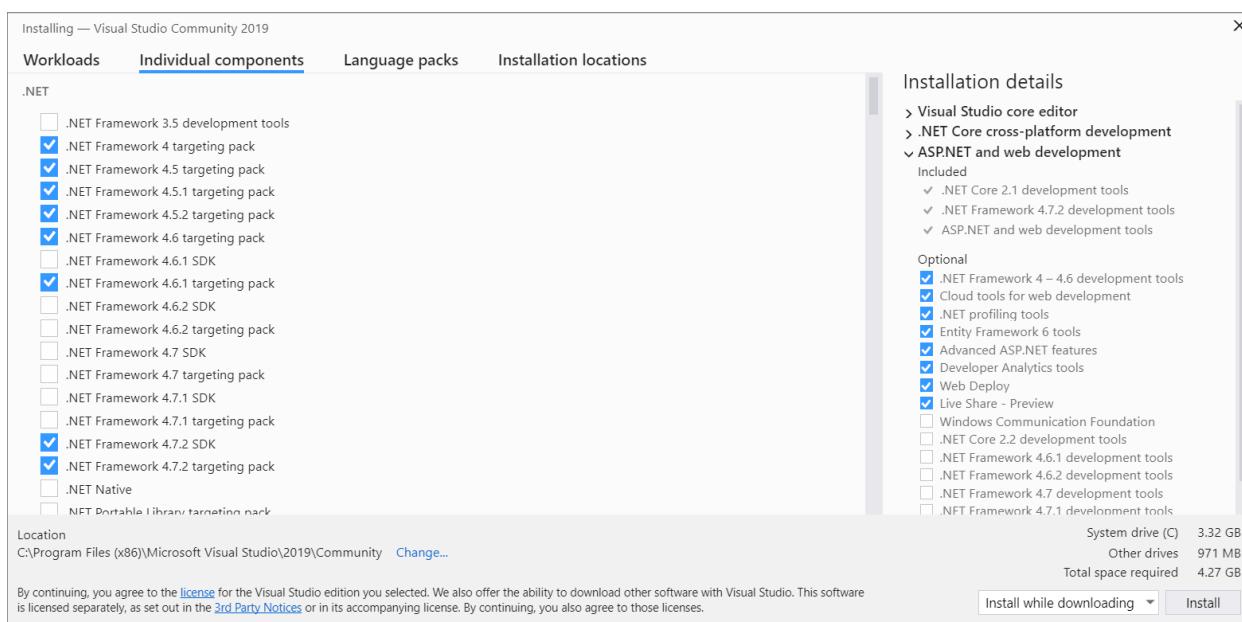
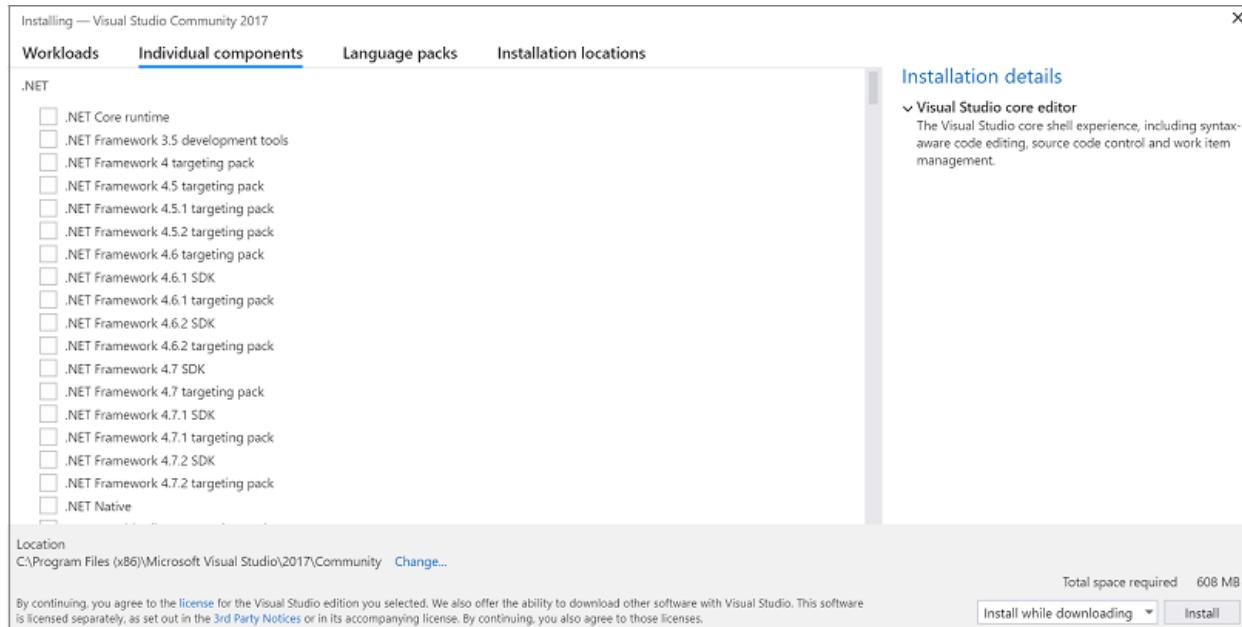
Next, status screens appear that show the progress of your Visual Studio installation.

TIP

At any time after installation, you can install workloads or components that you didn't install initially. If you have Visual Studio open, go to **Tools > Get Tools and Features...** which opens the Visual Studio Installer. Or, open **Visual Studio Installer** from the Start menu. From there, you can choose the workloads or components that you wish to install. Then, choose **Modify**.

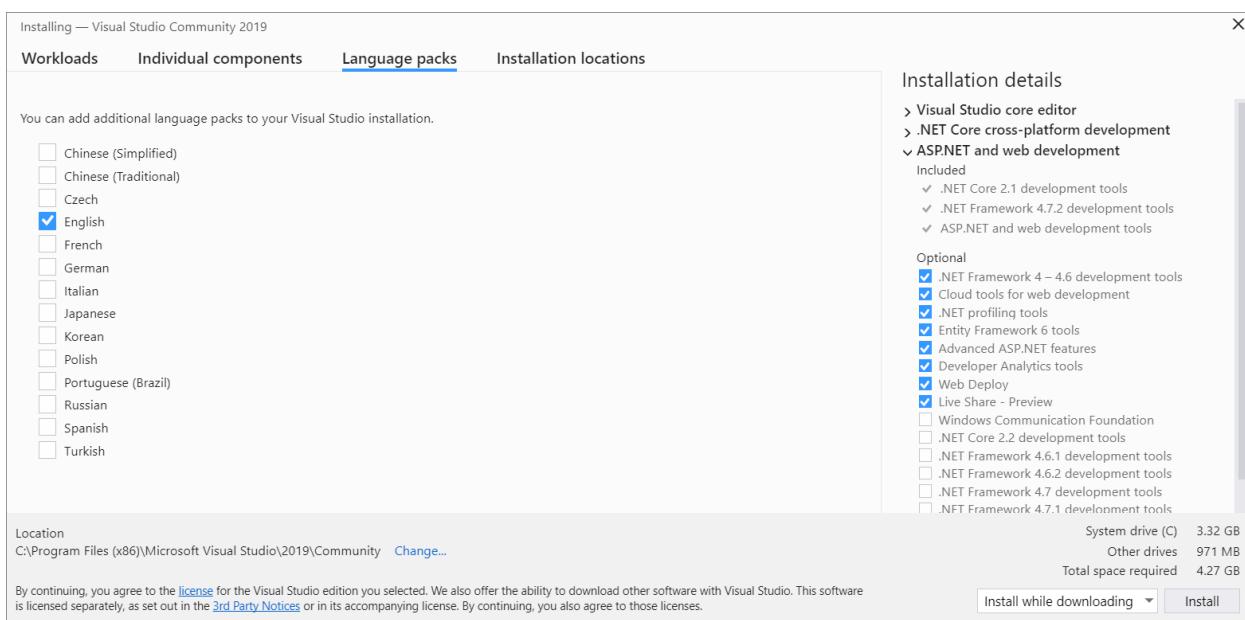
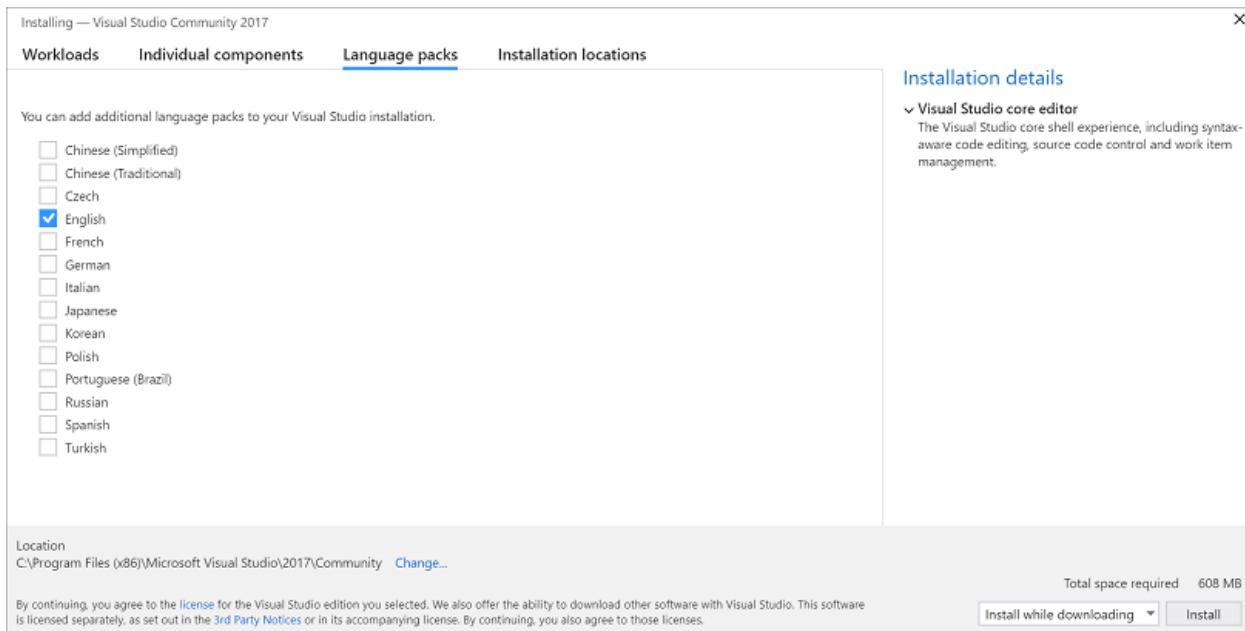
Step 5 - Choose individual components (Optional)

If you don't want to use the Workloads feature to customize your Visual Studio installation, or you want to add more components than a workload installs, you can do so by installing or adding individual components from the **Individual components** tab. Choose what you want, and then follow the prompts.



Step 6 - Install language packs (Optional)

By default, the installer program tries to match the language of the operating system when it runs for the first time. To install Visual Studio in a language of your choosing, choose the **Language packs** tab from the Visual Studio Installer, and then follow the prompts.



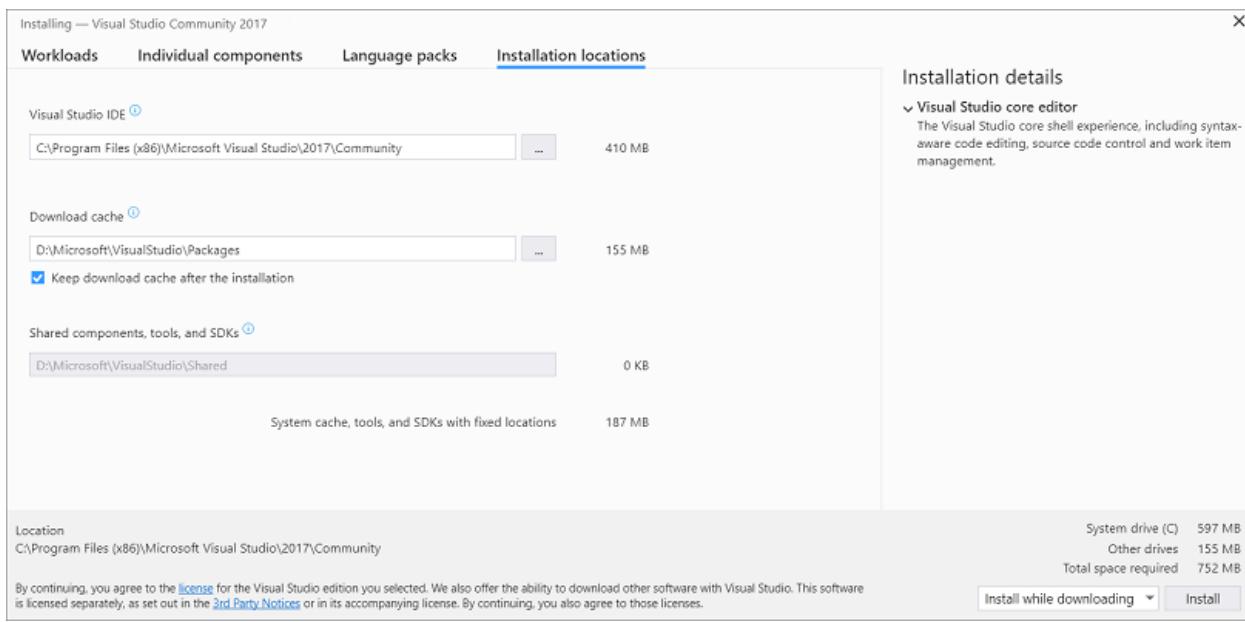
Change the installer language from the command line

Another way that you can change the default language is by running the installer from the command line. For example, you can force the installer to run in English by using the following command:

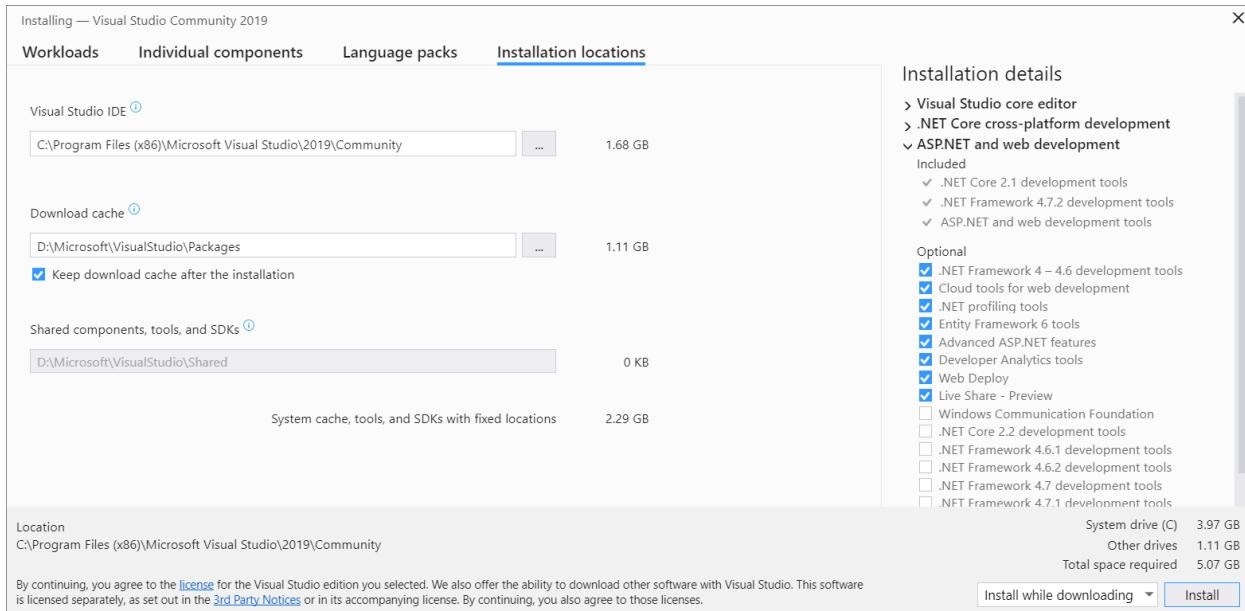
`vs_installer.exe --locale en-US`. The installer will remember this setting when it is run the next time. The installer supports the following language tokens: zh-cn, zh-tw, cs-cz, en-us, es-es, fr-fr, de-de, it-it, ja-jp, ko-kr, pl-pl, pt-br, ru-ru, and tr-tr.

Step 7 - Select the installation location (Optional)

New in 15.7: You can now reduce the installation footprint of Visual Studio on your system drive. You can choose to move the download cache, shared components, SDKs, and tools to different drives, and keep Visual Studio on the drive that runs it the fastest.



You can reduce the installation footprint of Visual Studio on your system drive. You can choose to move the download cache, shared components, SDKs, and tools to different drives, and keep Visual Studio on the drive that runs it the fastest.



IMPORTANT

You can select a different drive only when you first install Visual Studio. If you've already installed it and want to change drives, you must uninstall Visual Studio and then reinstall it.

For more information, see the [Select installation locations](#) page.

Step 8 - Start developing

1. After Visual Studio installation is complete, choose the **Launch** button to get started developing with Visual Studio.
2. Choose **File**, and then choose **New Project**.
3. Select a project type.

For example, to [build a C++ app](#), choose **Installed**, expand **Visual C++**, and then choose the C++

project type that you want to build.

To [build a C# app](#), choose **Installed**, expand **Visual C#**, and then choose the C# project type that you want to build.

1. After Visual Studio installation is complete, choose the **Launch** button to get started developing with Visual Studio.
2. On the start window, choose **Create a new project**.
3. In the search box, enter the type of app you want to create to see a list of available templates. The list of templates depends on the workload(s) that you chose during installation. To see different templates, choose different workloads.
You can also filter your search for a specific programming language by using the **Language** drop-down list. You can filter by using the **Platform** list and the **Project type** list, too.
4. Visual Studio opens your new project, and you're ready to code!

Get support

Sometimes, things can go wrong. If your Visual Studio installation fails, see [Troubleshoot Visual Studio installation and upgrade issues](#) for step-by-step guidance.

We also offer a [live chat](#) (English only) support option for installation-related issues.

Here are a few more support options:

- Report product issues to us via the [Report a Problem](#) tool that appears both in the Visual Studio Installer and in the Visual Studio IDE.
- Suggest a feature, track product issues, and find answers in the [Visual Studio Developer Community](#).
- Use your [GitHub](#) account to talk to us and other Visual Studio developers in the [Visual Studio conversation in the Gitter community](#).

See also

- [Update Visual Studio](#)
- [Modify Visual Studio](#)
- [Uninstall Visual Studio](#)
- [Create an offline installation of Visual Studio](#)
- [Use command-line parameters to install Visual Studio](#)
- [Install Visual Studio for Mac](#)

Create an offline installation of Visual Studio

10/23/2019 • 6 minutes to read • [Edit Online](#)

We designed Visual Studio 2017 to work well in a variety of network and computer configurations. While we recommend that you try the [Visual Studio web installer](#)—which is a small file and allows you to stay current with all the latest fixes and features—we understand that you might not be able to.

We designed Visual Studio 2019 to work well in a variety of network and computer configurations. While we recommend that you try the [Visual Studio web installer](#)—which is a small file and allows you to stay current with all the latest fixes and features—we understand that you might not be able to.

For example, you might have an unreliable internet connection or one that has low bandwidth. If so, you have a few options: You can use the new "Download all, then install" feature to download the files before you install, or you can use the command line to create a local cache of the files.

NOTE

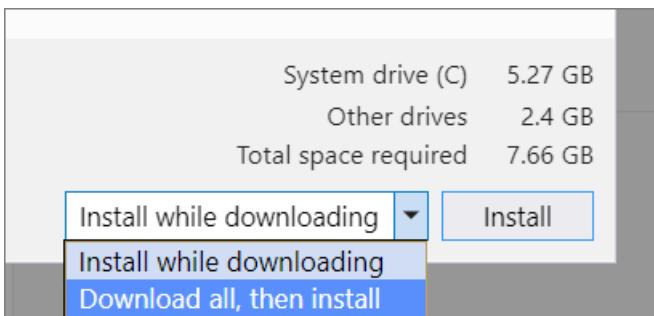
If you are an enterprise administrator who wants to perform a deployment of Visual Studio to a network of client workstations that are firewalled from the internet, see our [Create a network installation of Visual Studio](#) and [Install certificates required for Visual Studio offline installation](#) pages.

Use the "Download all, then install" feature

New in version 15.8: After you download the web installer, select the new **Download all, then install** option from the Visual Studio Installer. Then, continue with your installation.



After you download the web installer, select the new **Download all, then install** option from the Visual Studio Installer. Then, continue with your installation.



We designed the "Download all, then install" feature so that you can download Visual Studio as a single installation for the same computer on which you downloaded it. That way, you can safely disconnect from the web before you install Visual Studio.

IMPORTANT

Do not use the "Download all, then install" feature to create an offline cache that you intend to transfer to another computer. It's not designed to work that way.

If you want to create an offline cache to install Visual Studio on another computer, see the [Use the command line to create a local cache](#) section of this page for information about how to create a local cache, or the [Create a network installation of Visual Studio](#) page for information about how to create a network cache.

Use the command line to create a local cache

After you download a small bootstrapper, use the command line to create a local cache. Then, use the local cache to install Visual Studio. (This process replaces the ISO files that were available for previous versions.)

Here's how.

Step 1 - Download the Visual Studio bootstrapper

You must have an internet connection to complete this step.

To get a bootstrapper for Visual Studio 2017, see the [Visual Studio previous versions](#) download page for details on how to do so.

Your setup executable—or to be more specific, the bootstrapper file—should match or be similar to one of the following.

EDITION	FILENAME
Visual Studio Community	vs_community.exe
Visual Studio Professional	vs_professional.exe
Visual Studio Enterprise	vs_enterprise.exe
Visual Studio Build Tools	vs_buildtools.exe

Start by downloading the Visual Studio bootstrapper for your chosen edition of Visual Studio. Your setup file—or bootstrapper—will match or be similar to one of the following.

EDITION	FILE
Visual Studio Community	vs_community.exe
Visual Studio Professional	vs_professional.exe
Visual Studio Enterprise	vs_enterprise.exe
Visual Studio Build Tools	vs_buildtools.exe

TIP

If you previously downloaded a bootstrapper file and want to verify its version, here's how. In Windows, open File Explorer, right-click the bootstrapper file, choose **Properties**, choose the **Details** tab, and then view the **Product version** number. To match that number to a release of Visual Studio, see the [Visual Studio build numbers and release dates](#) page.

Step 2 - Create a local install cache

You must have an internet connection to complete this step.

IMPORTANT

If you install Visual Studio Community, you must activate it within 30 days of installation. This requires an internet connection.

Open a command prompt and use one of the commands from the following examples. The examples that are listed here assume that you're using the Community edition of Visual Studio; adjust the command as appropriate for your edition.

TIP

To prevent an error, make sure that your full installation path is less than 80 characters.

- For .NET web and .NET desktop development, run:

```
vs_community.exe --layout c:\vslayout --add Microsoft.VisualStudio.Workload.ManagedDesktop --add Microsoft.VisualStudio.Workload.NetWeb --add Component.GitHub.VisualStudio --includeOptional --lang en-US
```

- For .NET desktop and Office development, run:

```
vs_community.exe --layout c:\vslayout --add Microsoft.VisualStudio.Workload.ManagedDesktop --add Microsoft.VisualStudio.Workload.Office --includeOptional --lang en-US
```

- For C++ desktop development, run:

```
vs_community.exe --layout c:\vslayout --add Microsoft.VisualStudio.Workload.NativeDesktop --includeRecommended --lang en-US
```

- To create a complete local layout with all features (this will take a long time—we have *lots* of features!), run:

```
vs_community.exe --layout c:\vslayout --lang en-US
```

NOTE

A complete Visual Studio layout requires a minimum of 35 GB of disk space. For more information, see [System requirements](#). And for information about how to create a layout with only the components you want to install, see [Use command-line parameters to install Visual Studio](#).

NOTE

A complete Visual Studio layout requires a minimum of 35 GB of disk space. For more information, see [System requirements](#). And for information about how to create a layout with only the components you want to install, see [Use command-line parameters to install Visual Studio](#).

If you want to install a language other than English, change `en-US` to a locale from the [List of language locales](#).

Then, use the [list of the components and workloads available](#) to further customize your installation cache.

Step 3 - Install Visual Studio from the local cache

TIP

When you run from a local install cache, setup uses the local versions of each of these files. But if you select components during installation that aren't in the cache, setup attempts to download them from the internet.

IMPORTANT

For offline installations, if you get an error message that says "A product matching the following parameters cannot be found", make sure that you are using the `--noweb` switch with version 16.3.5 or later.

To make sure that you install only the files that you've previously downloaded, use the same command-line options that you used to create the layout cache. For example, if you created a layout cache with the following command:

```
vs_community.exe --layout c:\vslayout --add Microsoft.VisualStudio.Workload.ManagedDesktop --add  
Microsoft.VisualStudio.Workload.NetWeb --add Component.GitHub.VisualStudio --includeOptional --lang en-US
```

Then use this command to run the installation:

```
c:\vslayout\vs_community.exe --add Microsoft.VisualStudio.Workload.ManagedDesktop --add  
Microsoft.VisualStudio.Workload.NetWeb --add Component.GitHub.VisualStudio --includeOptional
```

For more examples of how to use [command-line parameters](#), see the [Command-line parameter examples for Visual Studio installation](#) page.

NOTE

If you get an error that a signature is invalid, you must install updated certificates. Open the Certificates folder in your offline cache. Double-click each of the certificate files, and then click through the Certificate Manager wizard. If you're asked for a password, leave it blank.

List of language locales

LANGUAGE-LOCALE	LANGUAGE
cs-CZ	Czech
de-DE	German
en-US	English
es-ES	Spanish
fr-FR	French
it-IT	Italian
ja-JP	Japanese

LANGUAGE-LOCALE	LANGUAGE
ko-KR	Korean
pl-PL	Polish
pt-BR	Portuguese - Brazil
ru-RU	Russian
tr-TR	Turkish
zh-CN	Chinese - Simplified
zh-TW	Chinese - Traditional

Get support

Sometimes, things can go wrong. If your Visual Studio installation fails, see [Troubleshoot Visual Studio installation and upgrade issues](#) for step-by-step guidance.

We also offer a [live chat](#) (English only) support option for installation-related issues.

Here are a few more support options:

- Report product issues to us via the [Report a Problem](#) tool that appears both in the Visual Studio Installer and in the Visual Studio IDE.
- Suggest a feature, track product issues, and find answers in the [Visual Studio Developer Community](#).
- Use your [GitHub](#) account to talk to us and other Visual Studio developers in the [Visual Studio conversation in the Gitter community](#).

See also

- [Create a network installation of Visual Studio](#)
- [Update a network-based installation of Visual Studio](#)
- [Install certificates required for Visual Studio offline installation](#)
- [Use command-line parameters to install Visual Studio](#)
- [Visual Studio workload and component IDs](#)

Install certificates required for Visual Studio offline installation

10/7/2019 • 7 minutes to read • [Edit Online](#)

Visual Studio is primarily designed to be installed on an internet-connected machine, since many components are updated regularly. However, with some extra steps, it's possible to deploy Visual Studio in an environment where a working internet connection is unavailable.

The Visual Studio setup engine installs only content that is trusted. It does this by checking Authenticode signatures of the content being downloaded and verifying that all content is trusted before installing it. This keeps your environment safe from attacks where the download location is compromised. Visual Studio setup therefore requires that several standard Microsoft root and intermediate certificates are installed and up-to-date on a user's machine. If the machine has been kept up to date with Windows Update, signing certificates usually are up to date. If the machine is connected to the internet, during installation Visual Studio may refresh certificates as necessary to verify file signatures. If the machine is offline, the certificates must be refreshed another way.

How to refresh certificates when offline

There are three options for installing or updating certificates in an offline environment.

Option 1 - Manually install certificates from a layout folder

When you create a network layout, the necessary certificates are downloaded to the Certificates folder. You can then manually install the certificates by double-clicking each of the certificate files, and then clicking through the Certificate Manager wizard. If asked for a password, leave it blank.

Update: For Visual Studio 2017 version 15.8 Preview 2 or later, you can manually install the certificates by right-clicking each of the certificate files, selecting Install Certificate, and then clicking through the Certificate Manager wizard.

When you create a network layout, the necessary certificates are downloaded to the Certificates folder. You can manually install the certificates by right-clicking each of the certificate files, selecting Install Certificate, and then clicking through the Certificate Manager wizard. If asked for a password, leave it blank.

Option 2 - Distribute trusted root certificates in an enterprise environment

For enterprises with offline machines that do not have the latest root certificates, an administrator can use the instructions on the [Configure Trusted Roots and Disallowed Certificates](#) page to update them.

Option 3 - Install certificates as part of a scripted deployment of Visual Studio

If you are scripting the deployment of Visual Studio in an offline environment to client workstations, you should follow these steps:

1. Copy the [Certificate Manager Tool](#) (certmgr.exe) to the installation share (for example, \\server\\share\\vs2017). Certmgr.exe is not included as part of Windows itself, but is available as part of the [Windows SDK](#).
2. Create a batch file with the following commands:

```
certmgr.exe -add -c certificates\manifestSignCertificates.p12 -n "Microsoft Code Signing PCA 2011" -s -r LocalMachine CA

certmgr.exe -add -c certificates\manifestSignCertificates.p12 -n "Microsoft Root Certificate Authority" -s -r LocalMachine root

certmgr.exe -add -c certificates\manifestCounterSignCertificates.p12 -n "Microsoft Time-Stamp PCA 2010" -s -r LocalMachine CA

certmgr.exe -add -c certificates\manifestCounterSignCertificates.p12 -n "Microsoft Root Certificate Authority" -s -r LocalMachine root

certmgr.exe -add -c certificates\vs_installer_opc.SignCertificates.p12 -n "Microsoft Code Signing PCA" -s -r LocalMachine CA

certmgr.exe -add -c certificates\vs_installer_opc.SignCertificates.p12 -n "Microsoft Root Certificate Authority" -s -r LocalMachine root
```

Update: For Visual Studio 2017 version 15.8 Preview 2 or later, create the batch file with the following commands:

```
certmgr.exe -add [layout path]\certificates\manifestRootCertificate.cer -n "Microsoft Root Certificate Authority 2011" -s -r LocalMachine root

certmgr.exe -add [layout path]\certificates\manifestCounterSignRootCertificate.cer -n "Microsoft Root Certificate Authority 2010" -s -r LocalMachine root

certmgr.exe -add [layout path]\certificates\vs_installer_opc.RootCertificate.cer -n "Microsoft Root Certificate Authority" -s -r LocalMachine root
```

Alternatively, create a batch file that uses certutil.exe, which ships with Windows, with the following commands:

```
certutil.exe -addstore -f "Root" "[layout path]\certificates\manifestRootCertificate.cer"

certutil.exe -addstore -f "Root" "[layout path]\certificates\manifestCounterSignRootCertificate.cer"

certutil.exe -addstore -f "Root" "[layout path]\certificates\vs_installer_opc.RootCertificate.cer"
```

3. Deploy the batch file to the client. This command should be run from an elevated process.
1. Copy the [Certificate Manager Tool](#) (certmgr.exe) to the installation share (for example, \\server\\share\\vs2019). Certmgr.exe is not included as part of Windows itself, but is available as part of the [Windows SDK](#).
2. Create a batch file with the following commands:

```
certmgr.exe -add [layout path]\certificates\manifestRootCertificate.cer -n "Microsoft Root Certificate Authority 2011" -s -r LocalMachine root

certmgr.exe -add [layout path]\certificates\manifestCounterSignRootCertificate.cer -n "Microsoft Root Certificate Authority 2010" -s -r LocalMachine root

certmgr.exe -add [layout path]\certificates\vs_installer_opc.RootCertificate.cer -n "Microsoft Root Certificate Authority" -s -r LocalMachine root
```

Alternatively, create a batch file that uses certutil.exe, which ships with Windows, with the following commands:

```
certutil.exe -addstore -f "Root" "[layout path]\certificates\manifestRootCertificate.cer"

certutil.exe -addstore -f "Root" "[layout path]\certificates\manifestCounterSignRootCertificate.cer"

certutil.exe -addstore -f "Root" "[layout path]\certificates\vs_installer_opc.RootCertificate.cer"
```

3. Deploy the batch file to the client. This command should be run from an elevated process.

What are the certificates files in the Certificates folder?

The three .P12 files in this folder each contain an intermediate certificate and a root certificate. Most systems that are current with Windows Update have these certificates already installed.

- **ManifestSignCertificates.p12** contains:
 - Intermediate certificate: **Microsoft Code Signing PCA 2011**
 - Not required. Improves performance in some scenarios if present.
 - Root certificate: **Microsoft Root Certificate Authority 2011**
 - Required on Windows 7 Service Pack 1 systems that do not have the latest Windows Updates installed.
- **ManifestCounterSignCertificates.p12** contains:
 - Intermediate certificate: **Microsoft Time-Stamp PCA 2010**
 - Not required. Improves performance in some scenarios if present.
 - Root certificate: **Microsoft Root Certificate Authority 2010**
 - Required for Windows 7 Service Pack 1 systems that do not have the latest Windows Updates installed.
- **Vs_installer_opc.SignCertificates.p12** contains:
 - Intermediate certificate: **Microsoft Code Signing PCA**
 - Required for all systems. Note that systems with all updates applied from Windows Update might not have this certificate.
 - Root certificate: **Microsoft Root Certificate Authority**
 - Required. This certificate ships with systems running Windows 7 or later.

Update: For Visual Studio 2017 version 15.8 Preview 2 or later, the Visual Studio Installer requires only the root certificates to be installed on the system. These certificates are stored in .cer files instead of .p12.

- **ManifestSignCertificates.cer** contains:
 - Root certificate: **Microsoft Root Certificate Authority 2011**
 - Required on Windows 7 Service Pack 1 systems that do not have the latest Windows Updates installed.
- **ManifestCounterSignCertificates.cer** contains:
 - Root certificate: **Microsoft Root Certificate Authority 2010**
 - Required for Windows 7 Service Pack 1 systems that do not have the latest Windows Updates installed.
- **Vs_installer_opc.SignCertificates.cer** contains:
 - Root certificate: **Microsoft Root Certificate Authority**
 - Required. This certificate ships with systems running Windows 7 or later.

The Visual Studio Installer requires only the root certificates to be installed on the system.

Why are the certificates from the Certificates folder not installed

automatically?

When a signature is verified in an online environment, Windows APIs are used to download and add the certificates to the system. Verification that the certificate is trusted and allowed via administrative settings occurs during this process. This verification process cannot occur in most offline environments. Installing the certificates manually allows enterprise administrators to ensure the certificates are trusted and meet the security policy of their organization.

Checking if certificates are already installed

One way to check on the installing system is to follow these steps:

1. Run **mmc.exe**.
 - a. Click **File**, and then select **Add/Remove Snap-in**.
 - b. Double-click **Certificates**, select **Computer account**, and then click **Next**.
 - c. Select **Local computer**, click **Finish**, and then click **OK**.
 - d. Expand **Certificates (Local Computer)**.
 - e. Expand **Trusted Root Certification Authorities**, and then select **Certificates**.
 - Check this list for the necessary root certificates.
 - f. Expand **Intermediate Certification Authorities**, and then select **Certificates**.
 - Check this list for the required intermediate certificates.
2. Click **File**, and then select **Add/Remove Snap-in**.
 - a. Double-click **Certificates**, select **My user account**, click **Finish**, and then click **OK**.
 - b. Expand **Certificates – Current User**.
 - c. Expand **Intermediate Certification Authorities**, and then select **Certificates**.
 - Check this list for the required intermediate certificates.

If the certificates names were not in the **Issued To** columns, they must be installed. If an intermediate certificate was only in the **Current User** Intermediate Certificate store, then it is available only to the user that is logged in. You might need to install it for other users.

Install Visual Studio

After you install the certificates, deployment of Visual Studio can proceed by using the instructions from the [Deploying from a network installation](#) section of the "Create a network installation of Visual Studio" page.

Get support

Sometimes, things can go wrong. If your Visual Studio installation fails, see [Troubleshoot Visual Studio installation and upgrade issues](#) for step-by-step guidance.

We also offer a [live chat](#) (English only) support option for installation-related issues.

Here are a few more support options:

- Report product issues to us via the [Report a Problem](#) tool that appears both in the Visual Studio Installer and in the Visual Studio IDE.
- Suggest a feature, track product issues, and find answers in the [Visual Studio Developer Community](#).
- Use your [GitHub](#) account to talk to us and other Visual Studio developers in the [Visual Studio conversation in the Gitter community](#).

See also

- [Install Visual Studio](#)
- [Visual Studio administrator guide](#)
- [Use command-line parameters to install Visual Studio](#)
- [Visual Studio workload and component IDs](#)

Install Visual Studio versions side-by-side

10/24/2019 • 3 minutes to read • [Edit Online](#)

You can install Visual Studio on a computer that has an earlier or later version of Visual Studio already installed.

Before you install versions side-by-side, review the following conditions:

- If you use Visual Studio 2017 to open a solution that was created in Visual Studio 2015, you can later open and modify the solution again in the earlier version as long as you haven't implemented any features that are specific to Visual Studio 2017.
- If you try to use Visual Studio 2017 to open a solution that was created in Visual Studio 2015 or an earlier version, you might need to modify your projects and files to be compatible with Visual Studio 2017. For more information, see the [Port, migrate, and upgrade Visual Studio Projects](#) page.

Before you install versions side-by-side, review the following conditions:

- If you use Visual Studio 2019 to open a solution that was created in Visual Studio 2017, you can later open and modify the solution again in the earlier version as long as you haven't implemented any features that are specific to Visual Studio 2019.
- If you try to use Visual Studio 2019 to open a solution that was created in Visual Studio 2017 or an earlier version, you might need to modify your projects and files to be compatible with Visual Studio 2019. For more information, see the [Port, migrate, and upgrade Visual Studio Projects](#) page.
- If you uninstall a version of Visual Studio on a computer that has more than one version installed, the file associations for Visual Studio are removed for all versions.
- Visual Studio doesn't automatically upgrade extensions because not all extensions are compatible. You must reinstall the extensions from the [Visual Studio Marketplace](#) or the software publisher.

.NET Framework versions and side-by-side installations

Visual Basic, Visual C#, and Visual F# projects use the **Target Framework** option in the **Project Designer** to specify which version of the .NET Framework that a project uses. For a C++ project, you can manually change the target framework by modifying the .vcxproj file. For more information, see the [Version compatibility in the .NET Framework](#) page.

When you create a project, you can specify which version of the .NET Framework the project targets in the **.NET Framework** list in the **New Project** dialog box.

For language-specific information, see the appropriate topic in the following table.

LANGUAGE	TOPIC
Visual Basic	Application Page, Project Designer (Visual Basic)
Visual C#	Application Page, Project Designer (C#)
Visual F#	Develop with Visual F# in Visual Studio
C++	How to: Modify the target framework and platform toolset

Get support

Sometimes, things can go wrong. If your Visual Studio installation fails, see [Troubleshoot Visual Studio installation and upgrade issues](#) for step-by-step guidance.

We also offer a [live chat](#) (English only) support option for installation-related issues.

Here are a few more support options:

- Report product issues to us via the [Report a Problem](#) tool that appears both in the Visual Studio Installer and in the Visual Studio IDE.
- Suggest a feature, track product issues, and find answers in the [Visual Studio Developer Community](#).
- Use your [GitHub](#) account to talk to us and other Visual Studio developers in the [Visual Studio conversation in the Gitter community](#).

See also

- [Install Visual Studio](#)
- [Port, migrate, and upgrade Visual Studio projects](#)
- [Building C/C++ isolated applications and side-by-side assemblies](#)

LANGUAGE	TOPIC
Visual Basic	Application Page, Project Designer (Visual Basic)
Visual C#	Application Page, Project Designer (C#)
Visual F#	Develop with Visual F# in Visual Studio
C++	How to: Modify the target framework and platform toolset

Get support

Sometimes, things can go wrong. If your Visual Studio installation fails, see [Troubleshoot Visual Studio installation and upgrade issues](#) for step-by-step guidance.

We also offer a [live chat](#) (English only) support option for installation-related issues.

Here are a few more support options:

- Report product issues to us via the [Report a Problem](#) tool that appears both in the Visual Studio Installer and in the Visual Studio IDE.
- Suggest a feature, track product issues, and find answers in the [Visual Studio Developer Community](#).
- Use your [GitHub](#) account to talk to us and other Visual Studio developers in the [Visual Studio conversation in the Gitter community](#).

See also

- [Install Visual Studio](#)
- [Port, migrate, and upgrade Visual Studio projects](#)
- [Building C/C++ isolated applications and side-by-side assemblies](#)

Select the installation locations in Visual Studio

4/2/2019 • 3 minutes to read • [Edit Online](#)

You can reduce the installation footprint of Visual Studio on your system drive by changing the location for some of its files. Specifically, you can use a different location for the download cache, shared components, SDKs, and tools files.

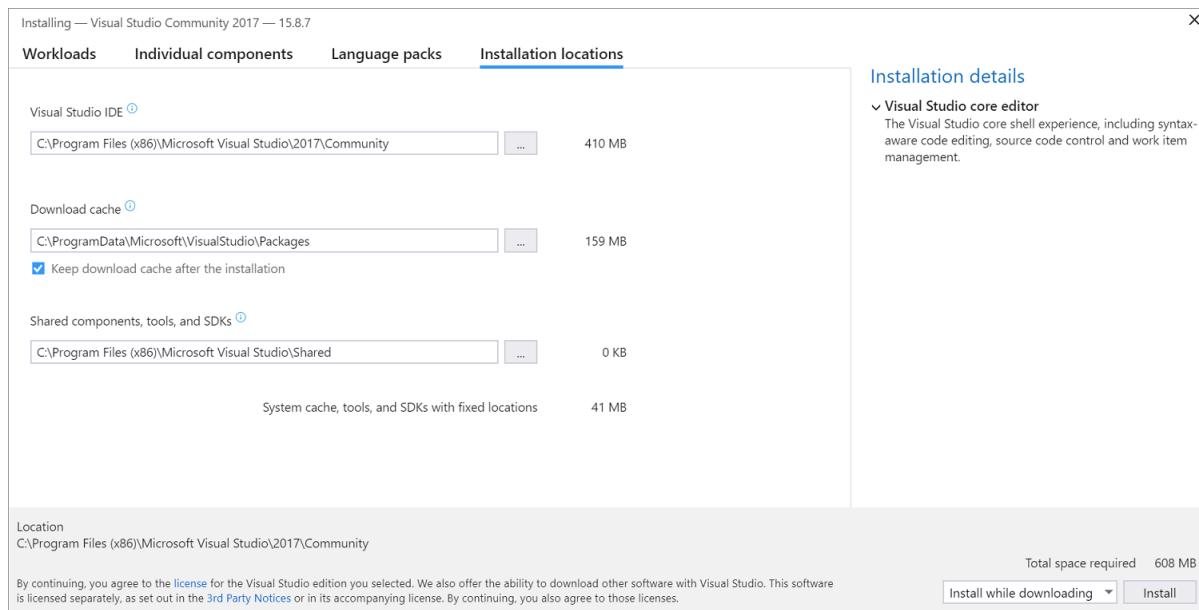
New in version 15.7: You can reduce the installation footprint of Visual Studio on your system drive by changing the location for some of its files. Specifically, you can use a different location for the download cache, shared components, SDKs, and tools files.

NOTE

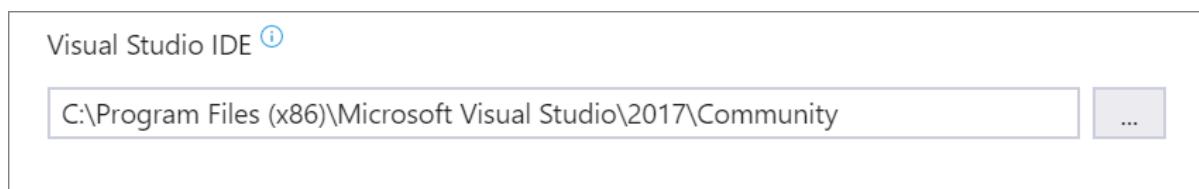
There are some tools and SDKs that have different rules on where they can be installed. Such tools and SDKs are installed on your system drive even if you choose another location.

Ready to get started? Here's how.

1. When you install Visual Studio, choose the **Installation locations** tab.



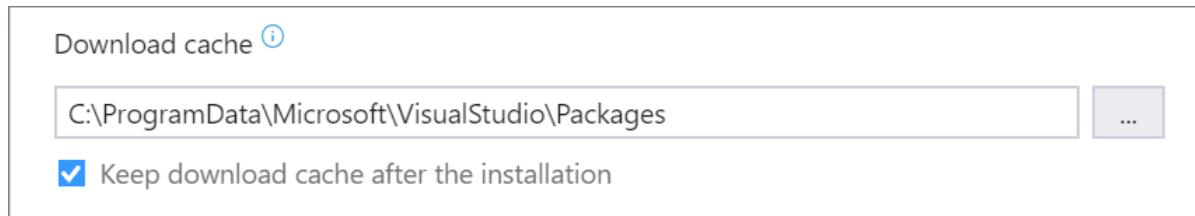
2. In the **Visual Studio IDE** section, accept the default. Visual Studio installs the core product and includes files that are specific to this version of Visual Studio.



TIP

If your system drive is a solid-state drive (SSD), we recommend that you accept the default location on your system drive. The reason? When you develop with Visual Studio, you read from and write to a lot of files, which increases the disk I/O activity. It's best to choose your fastest drive to handle the load.

3. In the **Download cache** section, decide if you want to keep the download cache, and then decide where you want to store its files.



- a. Check or uncheck **Keep download cache after the installation**.

If you decide not to keep the download cache, the location is used only temporarily. This action won't affect or delete files from previous installations.

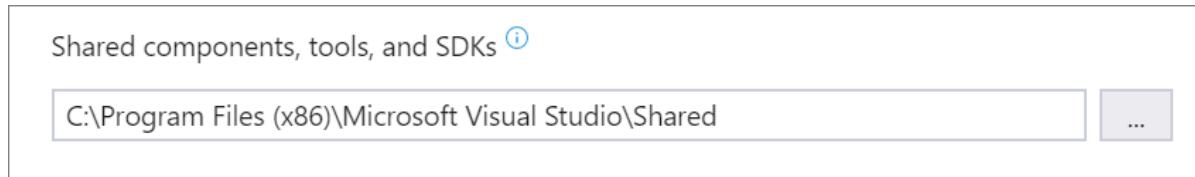
- b. Specify the drive where you want to store installation files and manifests from the download cache.

For example, if you select the "Desktop development with C++" workload, the temporarily required size is 1.58 GB on your system drive, which is then freed as soon as the installation completes.

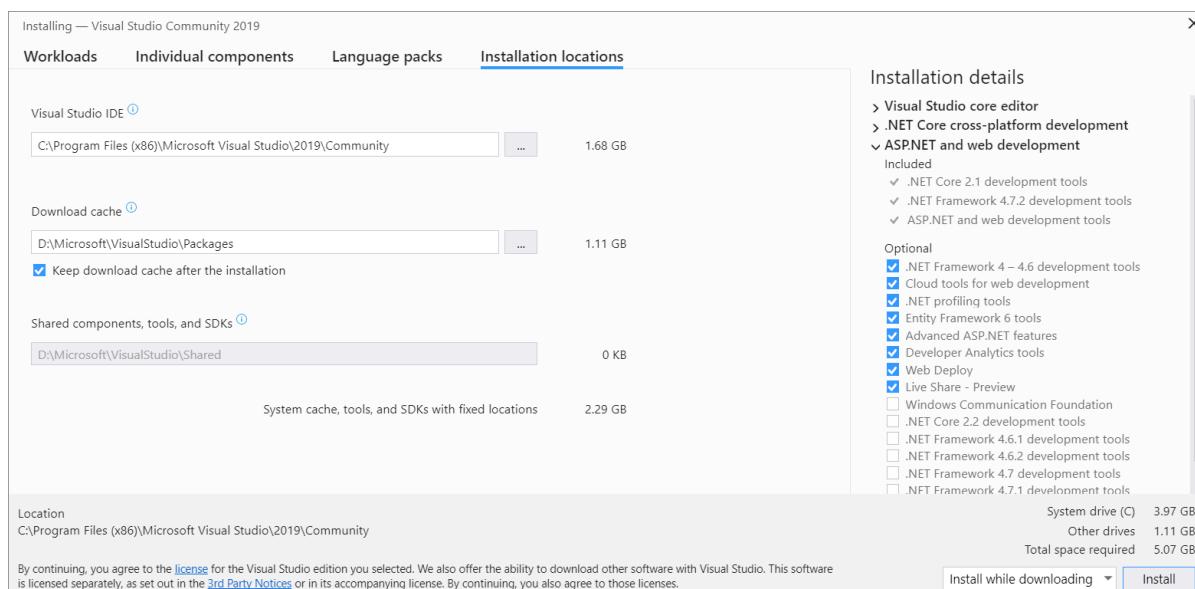
IMPORTANT

This location is set with your first installation and cannot be changed later from the installer UI. Instead, you must [use command-line parameters](#) to move the download cache.

4. In the **Shared components, tools, and SDKs** section, specify the drive where you want to store the files that are shared by side-by-side Visual Studio installations. SDKs and tools are also stored in this directory.



1. When you install Visual Studio, choose the **Installation locations** tab.



2. In the **Visual Studio IDE** section, accept the default. Visual Studio installs the core product and includes files that are specific to this version of Visual Studio.

TIP

If your system drive is a solid-state drive (SSD), we recommend that you accept the default location on your system drive. The reason? When you develop with Visual Studio, you read from and write to a lot of files, which increases the disk I/O activity. It's best to choose your fastest drive to handle the load.

3. In the **Download cache** section, decide if you want to keep the download cache, and then decide where you want to store its files.

- Check or uncheck **Keep download cache after the installation.**

If you decide not to keep the download cache, the location is used only temporarily. This action won't affect or delete files from previous installations.

- Specify the drive where you want to store installation files and manifests from the download cache.

For example, if you select the "Desktop development with C++" workload, the temporarily required size is 1.58 GB on your system drive, which is then freed as soon as the installation completes.

IMPORTANT

This location is set with your first installation and cannot be changed later from the installer UI. Instead, you must [use command-line parameters](#) to move the download cache.

4. In the **Shared components, tools, and SDKs** section, note that it uses the same drive that you chose in the "Download cache" section. The \Microsoft\VisualStudio\Shared directory is where Visual Studio stores the files that are shared by side-by-side Visual Studio installations. SDKs and tools are also stored in this directory.

Get support

Sometimes, things can go wrong. If your Visual Studio installation fails, see [Troubleshoot Visual Studio installation and upgrade issues](#) for step-by-step guidance.

We also offer a [live chat](#) (English only) support option for installation-related issues.

Here are a few more support options:

- Report product issues to us via the [Report a Problem](#) tool that appears both in the Visual Studio Installer and in the Visual Studio IDE.
- Suggest a feature, track product issues, and find answers in the [Visual Studio Developer Community](#).
- Use your [GitHub](#) account to talk to us and other Visual Studio developers in the [Visual Studio conversation in the Gitter community](#).

See also

- [Install Visual Studio](#)
- [Update Visual Studio](#)
- [Modify Visual Studio](#)
- [Uninstall Visual Studio](#)

Import or export installation configurations

5/17/2019 • 2 minutes to read • [Edit Online](#)

You can configure Visual Studio across your organization with installation configuration files. To do so, simply export the workload and component information to a .vsconfig file by using the Visual Studio installer. You can then import the configuration into new or existing installations, and share them with others, too.

Here's how.

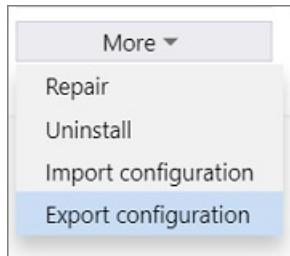
NOTE

This functionality is available only in Visual Studio 2017 version 15.9 and later.

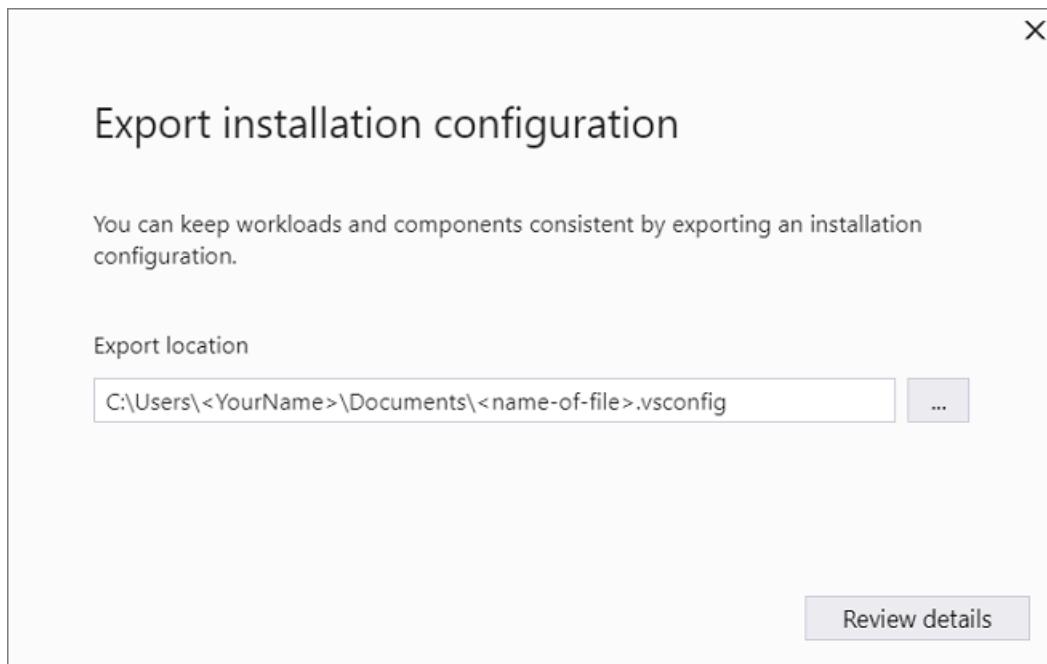
Export a configuration

You can choose to export an installation configuration file from either a previously installed instance of Visual Studio or one that you're currently installing.

1. Open the Visual Studio Installer.
2. On the product card, choose the **More** button, and then select **Export configuration**.



3. Browse to or type the location where you want to save your .vsconfig file, and then choose **Review details**.



4. Make sure you've got the workloads and components that you want, and then choose **Export**.

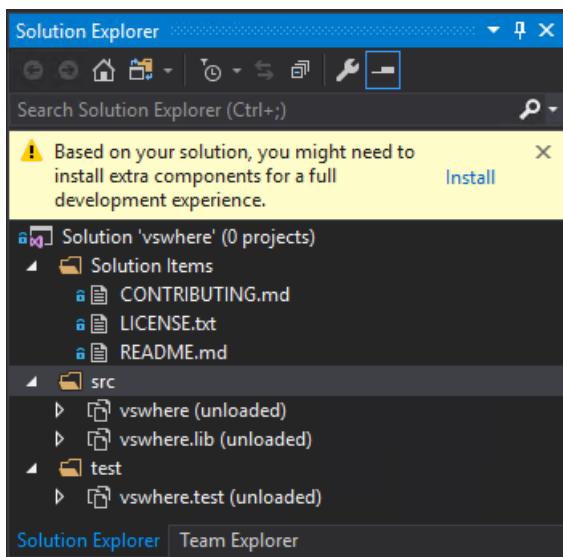
Import a configuration

When you're ready to import an installation configuration file, follow these steps.

1. Open the Visual Studio Installer.
2. On the product card, choose the **More** button, and then select **Import configuration**.
3. Locate the .vsconfig file that you want to import, and then choose **Review details**.
4. Make sure you've got the workloads and components that you want, and then choose **Close**.

Automatically install missing components

New in Visual Studio 2019: When you save a .vsconfig file to your solution root directory and then open a solution, Visual Studio automatically detects which components are missing and prompts you to install them.



You can also generate a .vsconfig file right from Solution Explorer.

1. Right-click on your solution file.
2. Choose **Add > Installation Configuration File**.
3. Confirm the location where you want to save the .vsconfig file, and then choose **Review details**.
4. Make sure you've got the workloads and components that you want, and then choose **Export**.

NOTE

For more information, see the [Configure Visual Studio across your organization with .vsconfig](#) blog post.

Get support

Sometimes, things can go wrong. If your Visual Studio installation fails, see [Troubleshoot Visual Studio installation and upgrade issues](#) for step-by-step guidance.

We also offer a [live chat](#) (English only) support option for installation-related issues.

Here are a few more support options:

- Report product issues to us via the [Report a Problem](#) tool that appears both in the Visual Studio Installer and in the Visual Studio IDE.

- Suggest a feature, track product issues, and find answers in the [Visual Studio Developer Community](#).
- Use your [GitHub](#) account to talk to us and other Visual Studio developers in the [Visual Studio conversation in the Gitter community](#).

See also

- [Create a network installation of Visual Studio](#)
- [Update a networked-based installation of Visual Studio](#)
- [Control updates to Visual Studio deployments](#)
- [Set defaults for enterprise deployments](#)

Troubleshoot Visual Studio installation and upgrade issues

9/17/2019 • 5 minutes to read • [Edit Online](#)

IMPORTANT

Having a problem installing? We can help. We offer a [live chat](#) (English only) support option.

This troubleshooting guide includes step-by-step instructions that should resolve most installation issues.

Online installations

The following steps are optimized for a typical online installation. For an issue that affects an offline installation, please see [How to troubleshoot an offline installation](#).

Step 1 - Check whether this problem is a known issue

There are some known issues with the Visual Studio Installer that Microsoft is working on fixing. To see if there's a workaround for your problem, check the [Known Issues section of our release notes](#).

There are some known issues with the Visual Studio Installer that Microsoft is working on fixing. To see if there's a workaround for your problem, check the [Known Issues section of our release notes](#).

Step 2 - Check with the developer community

Search on your error message with the [Visual Studio Developer Community](#). Other members of the community might have documented a solution to your problem.

Step 3 - Delete the Visual Studio Installer directory to fix upgrade problems

The Visual Studio Installer bootstrapper is a minimal light-weight executable that installs the rest of the Visual Studio Installer. Deleting Visual Studio Installer files and then rerunning the bootstrapper might solve some update failures.

NOTE

Performing the following actions reinstalls the Visual Studio Installer files and resets the installation metadata.

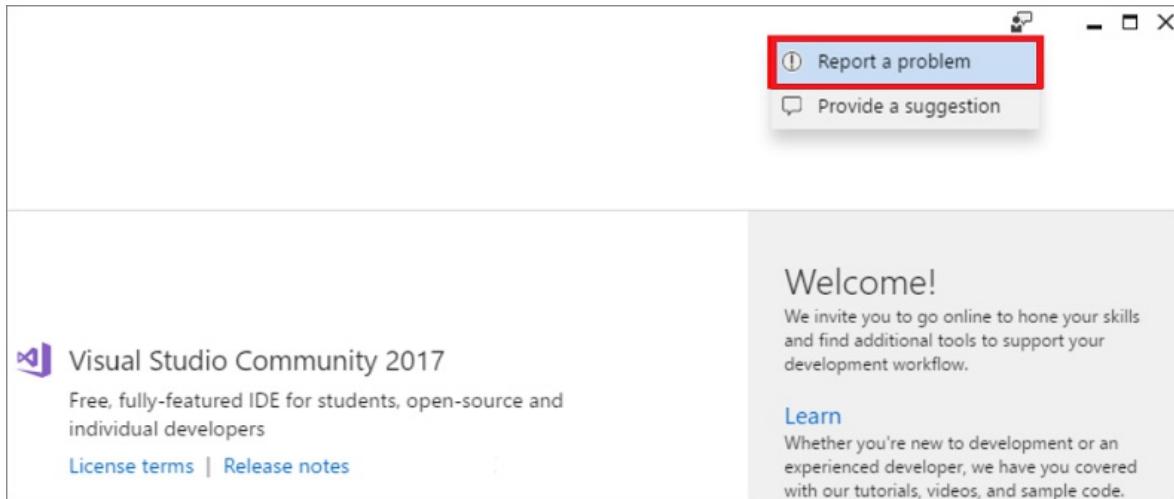
1. Close the Visual Studio Installer.
2. Delete the Visual Studio Installer directory. Typically, the directory is
`C:\Program Files (x86)\Microsoft Visual Studio\Installer`.
3. Run the Visual Studio Installer bootstrapper. You might find the bootstrapper in your Downloads folder with a file name that follows a `vs_[Visual Studio edition]_*_.exe` pattern. If you don't find that application, you can download the bootstrapper by going to the [Visual Studio downloads](#) page and clicking **Download** for your edition of Visual Studio. Then, run the executable to reset your installation metadata.
4. Try to install or update Visual Studio again. If the Installer continues to fail, go to the next step.
 1. Close the Visual Studio Installer.
 2. Delete the Visual Studio Installer directory. Typically, the directory is
`C:\Program Files (x86)\Microsoft Visual Studio\Installer`.

3. Run the Visual Studio Installer bootstrapper. You might find the bootstrapper in your Downloads folder with a file name that follows a `vs_[Visual Studio edition]_*.exe` pattern. If you don't find that application, you can download the bootstrapper by going to the [Visual Studio downloads](#) page and clicking **Download** for your edition of Visual Studio. Then, run the executable to reset your installation metadata.
4. Try to install or update Visual Studio again. If the Installer continues to fail, go to the next step.

Step 4 - Report a problem

In some situations, such as those related to corrupted files, the problems might have to be looked at on a case-by-case basis. To help us help you, please do the following:

1. Collect your setup logs. See [How to get the Visual Studio installation logs](#) for details.
2. Open the Visual Studio Installer, and then click **Report a problem** to open the Visual Studio Feedback tool.



3. Give your problem report a title, and provide relevant details. Click **Next** to go to the **Attachments** section, and then attach the generated log file (typically, the file is at `%TEMP%\vslogs.zip`).
4. Click **Next** to review your problem report, and then click **Submit**.

 1. Collect your setup logs. See [How to get the Visual Studio installation logs](#) for details.
 2. Open the Visual Studio Installer, and then click **Report a problem** to open the Visual Studio Feedback tool.



3. Give your problem report a title, and provide relevant details. Click **Next** to go to the **Attachments** section, and then attach the generated log file (typically, the file is at `%TEMP%\vslogs.zip`).
4. Click **Next** to review your problem report, and then click **Submit**.

Step 5 - Run `InstallCleanup.exe` to remove installation files

As a last resort, you can [remove Visual Studio](#) to remove all installation files and product information.

1. Follow the instructions in [Remove Visual Studio](#).
2. Rerun the bootstrapper that's described in [Step 3 - Delete the Visual Studio Installer directory to fix upgrade problems](#).
3. Try to install or update Visual Studio again.

Step 6 - Contact us (optional)

If none of the previous steps help you successfully install or upgrade Visual Studio, contact us by using our [live chat](#) support option (English only) for further assistance.

Offline installations

Here is a table of known issues and some workarounds that might help you when you create an [offline installation](#) and then install from a local layout.

ISSUE	ITEM	SOLUTION
Users do not have access to files.	permissions (ACLs)	Make sure that you adjust the permissions (ACLs) so that they grant Read access to other users <i>before</i> you share the offline install.
New workloads, components, or languages fail to install.	<code>--layout</code>	Make sure that you have internet access if you install from a partial layout and select workloads, components, or languages that were not downloaded previously in that partial layout.

For more information about how to resolve issues with a [network installation](#), see [Troubleshoot network-related errors when you install or use Visual Studio](#).

Installation logs

Setup logs are needed to troubleshoot most installation issues. When you submit an issue by using [Report a Problem](#) in the Visual Studio Installer, these logs are automatically included in your report.

If you contact Microsoft Support, you might need to provide these setup logs by using the [Microsoft Visual Studio and .NET Framework Log Collection Tool](#). The log collection tool collects setup logs from all components installed by Visual Studio, including .NET Framework, Windows SDK, and SQL Server. It also collects computer information, a Windows Installer inventory, and Windows event log information for Visual Studio Installer, Windows Installer, and System Restore.

To collect the logs:

1. [Download the tool](#).
2. Open an administrative command prompt.
3. Run `collect.exe` from the directory where you saved the tool.
4. Find the resulting `vslogs.zip` file in your `%TEMP%` directory, for example,
`C:\Users\YourName\AppData\Local\Temp\vslogs.zip`.

NOTE

The tool must be run under the same user account that the failed installation was run under. If you are running the tool from a different user account, set the `-user:<name>` option to specify the user account under which the failed installation was run. Run `Collect.exe -?` from an administrator command prompt for additional options and usage information.

Live help

If the solutions listed in this troubleshooting guide do not help you to successfully install or upgrade Visual Studio, use our [live chat](#) support option (English only) for further assistance.

See also

- [Remove Visual Studio](#)
- [Install and use Visual Studio and Azure Services behind a firewall or proxy server](#)
- [Tools for detecting and managing Visual Studio instances](#)
- [Visual Studio administrator guide](#)

Update Visual Studio to the most recent release

9/4/2019 • 7 minutes to read • [Edit Online](#)

We encourage you to update to the most [recent release](#) of Visual Studio 2017 so that you always get the latest features, fixes, and improvements.

And if you'd like to try out our newest version, consider downloading and installing [Visual Studio 2019](#) instead.

IMPORTANT

You must log on with an account that has administrative permissions to install, update, or modify Visual Studio. For more information, see [User Permissions and Visual Studio](#).

NOTE

This topic applies to Visual Studio on Windows. For Visual Studio for Mac, see [Update Visual Studio for Mac](#).

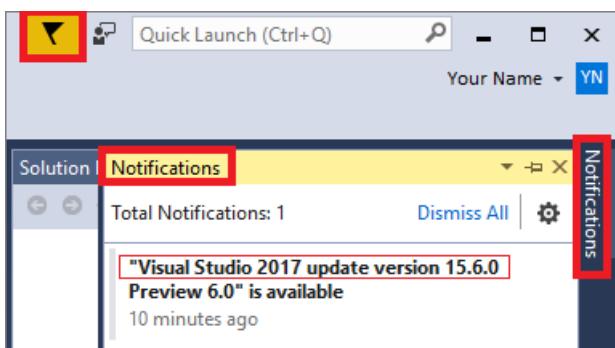
Update Visual Studio 2017 version 15.6 or later

We've streamlined the installation and update experience to make it easier to use directly from within the IDE. Here's how to update from version 15.6 and later to newer versions of Visual Studio.

Using the Notifications hub

When there's an update, there's a corresponding notification flag in Visual Studio.

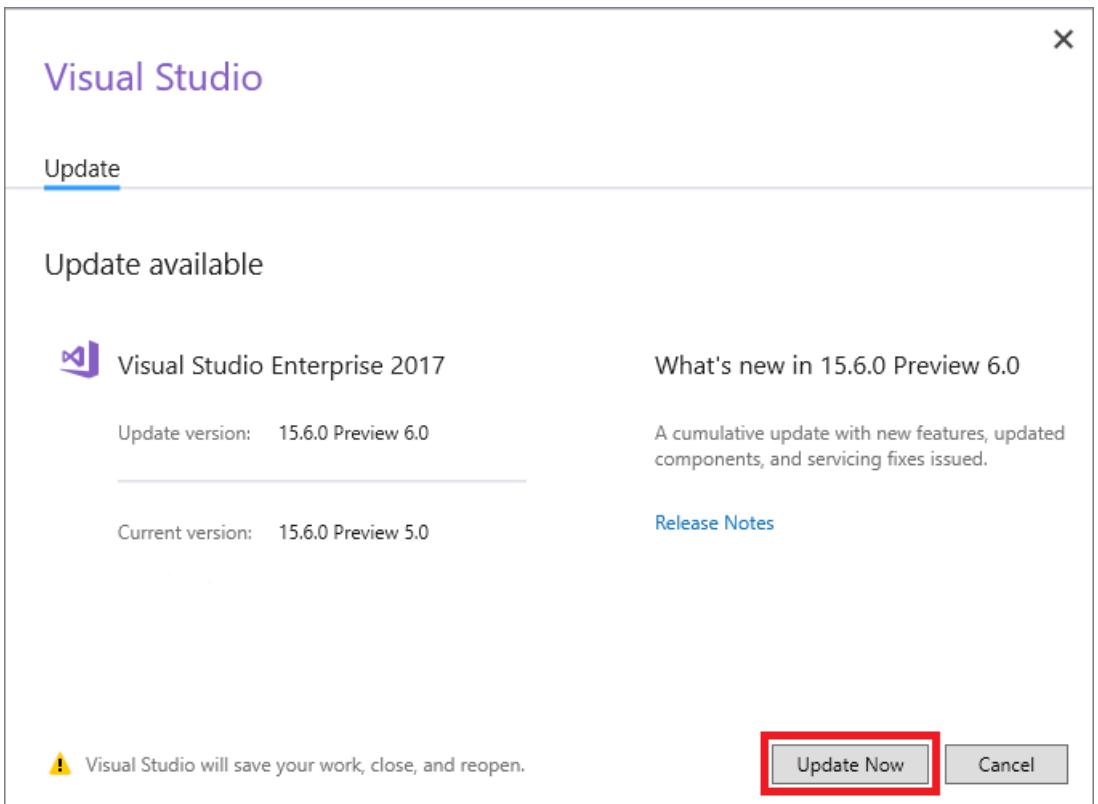
1. Save your work.
2. Choose the notification flag to open the **Notifications** hub, and then choose the update that you want to install.



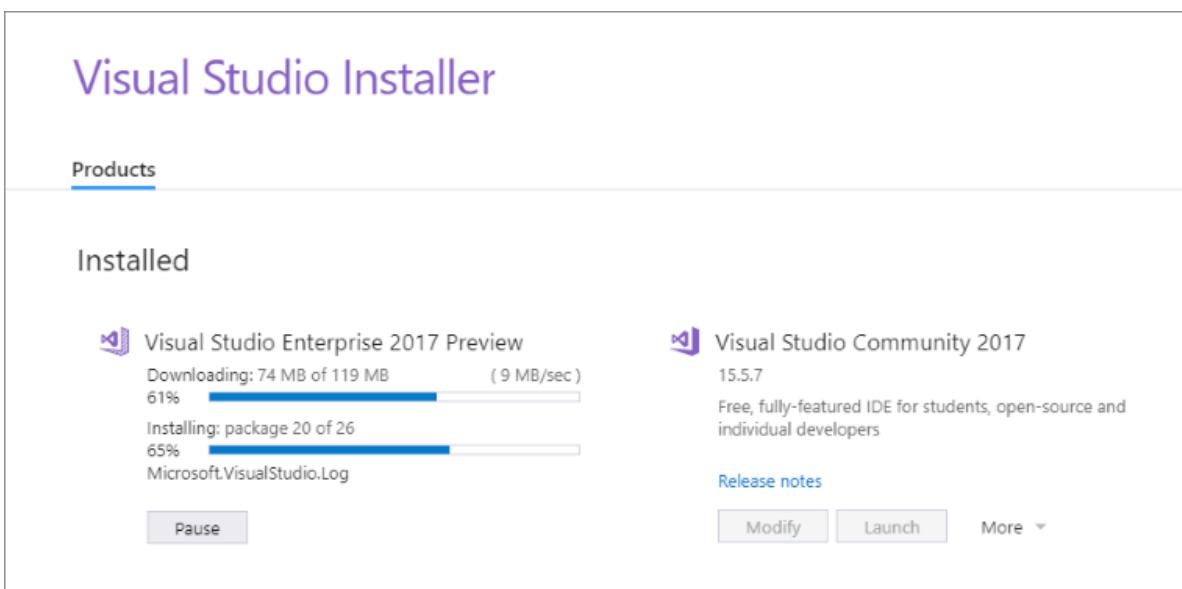
TIP

An update for an edition of Visual Studio 2017 is cumulative, so always choose to install the one with the most recent version number.

3. When the **Update** dialog box opens, choose **Update Now**.



If a User Access Control dialog box opens, choose **Yes**. Next, a "Please wait" dialog might open for a moment, and then the Visual Studio Installer opens to start the update.



Your update continues. Then, when it's complete, Visual Studio restarts.

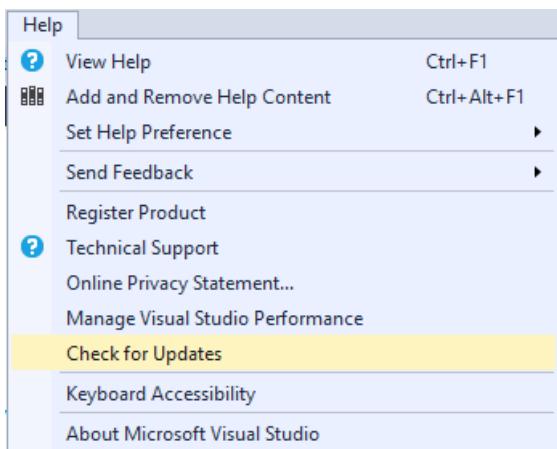
NOTE

When you run Visual Studio in administrator mode, you must manually restart Visual Studio after the update.

Using the IDE

You can check for an update and then install the update from the menu bar in Visual Studio.

1. Save your work.
2. Choose **Help > Check for Updates**.



3. When the **Update** dialog box opens, choose **Update Now**.

The update proceeds as described in the previous section, and then Visual Studio restarts after the update completes successfully.

NOTE

When you run Visual Studio in administrator mode, you must manually restart Visual Studio after the update.

Using the Visual Studio Installer

As in earlier versions of Visual Studio, you can use the Visual Studio Installer to install an update.

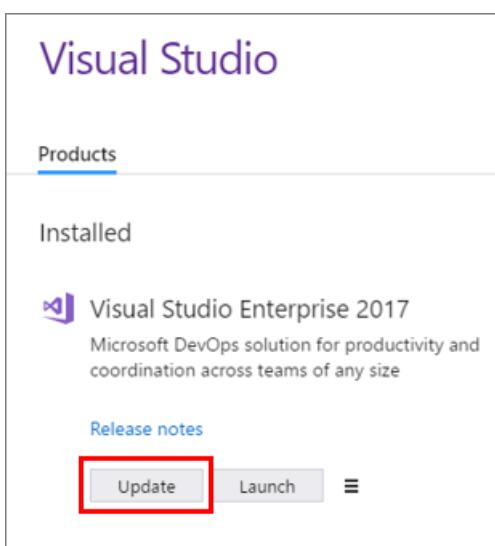
1. Save your work.
2. Open the installer. The Visual Studio Installer might require updating before you continue.

NOTE

On a computer running Windows 10, you can find the installer under the letter **V** as the **Visual Studio Installer**, or under the letter **M** as the **Microsoft Visual Studio Installer**.

3. On the **Product** page in the installer, look for the edition of Visual Studio that you installed previously.
4. If an update is available, you see an **Update** button. (It might take a few seconds for the installer to determine whether an update is available.)

Choose the **Update** button to install the updates.



Update Visual Studio 2017 version 15.5 or earlier

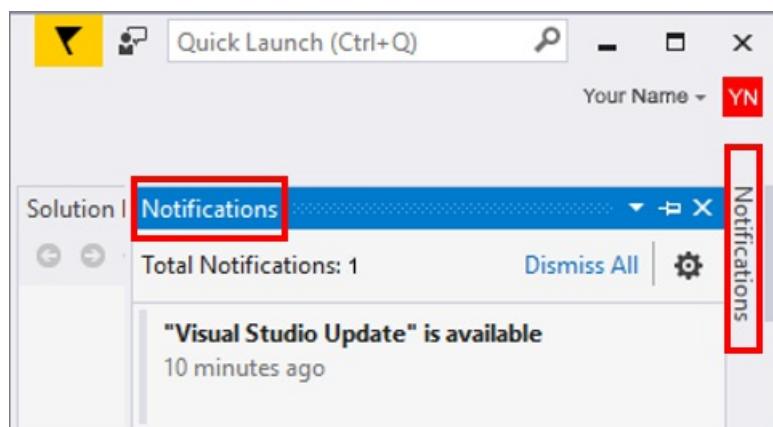
If you're using an earlier version, here's how to apply an update from Visual Studio 2017 version 15.0 through version 15.5.

Update by using the Notifications hub

- When there are updates, there's a corresponding notification flag in Visual Studio.



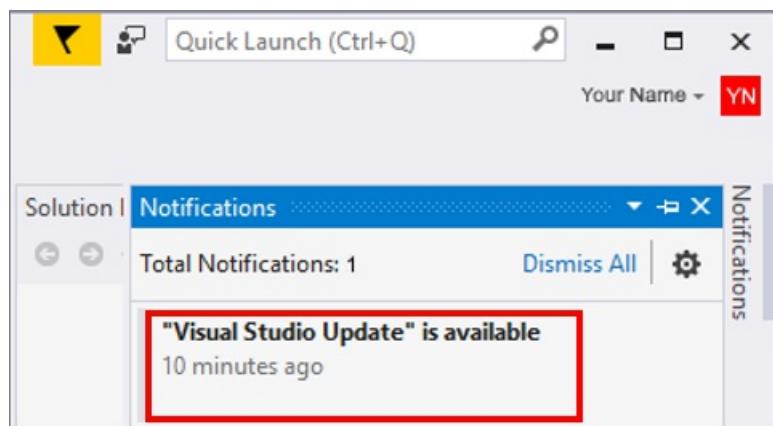
Choose the notification flag to open the **Notifications** hub.



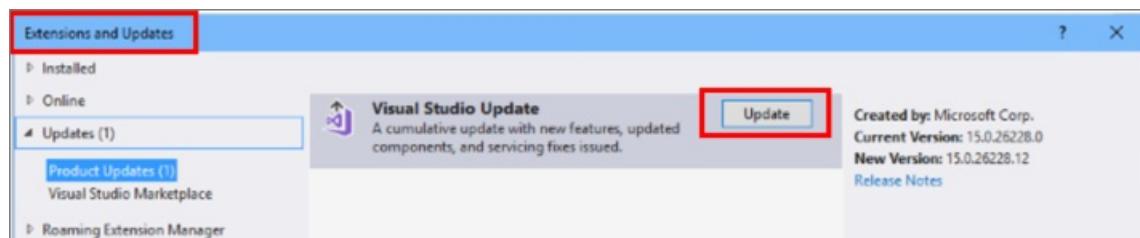
TIP

An update for an edition of Visual Studio 2017 is cumulative, so always choose to install the one with the most recent version number.

- Choose "**Visual Studio Update**" is available, which opens the **Extensions and Updates** dialog box.



- In the **Extensions and Updates** dialog box, choose the **Update** button.



More about Visual Studio notifications

Visual Studio notifies you when an update is available for Visual Studio itself or for any components, and also when certain events occur in the Visual Studio environment.

- When the notification flag is yellow, there's a Visual Studio product update available for you to install.
- When the notification flag is red, there's a problem with your license.
- When the notification flag is black, there are optional or informational messages to review.

Choose the notifications flag to open the **Notifications** hub and then choose the notifications that you want to act on. Or, choose to ignore or dismiss a notification.



If you choose to ignore a notification, Visual Studio stops showing it. If you want to reset the list of ignored notifications, choose the **Settings** button in the Notifications hub.



Update by using the Visual Studio Installer

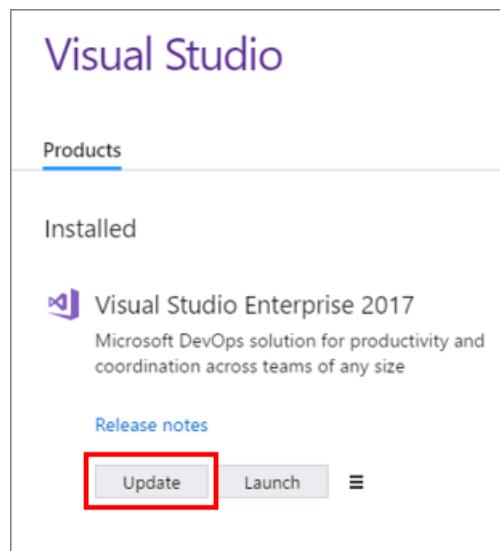
1. Open the installer. You might need to update the installer before continuing. If this is the case, you're prompted to do so.

NOTE

On a computer running Windows 10, you can find the installer under the letter **V** as the **Visual Studio Installer**, or under the letter **M** as the **Microsoft Visual Studio Installer**.

2. On the **Product** page in the installer, look for the edition of Visual Studio that installed previously.
3. If an update is available, you see an **Update** button. (It might take a few seconds for the installer to determine whether an update is available.)

Choose the **Update** button to install the updates.



We encourage you to update to the most [recent release](#) of Visual Studio 2019 so that you always get the latest features, fixes, and improvements.

And if you haven't already installed Visual Studio 2019, go to the [Visual Studio downloads](#) page to install it for free.

IMPORTANT

You must log on with an account that has administrative permissions to install, update, or modify Visual Studio. For more information, see [User Permissions and Visual Studio](#).

NOTE

This topic applies to Visual Studio on Windows. For Visual Studio for Mac, see [Update Visual Studio for Mac](#).

Here's how to update Visual Studio 2019.

Use the Visual Studio Installer

1. Open the installer.



You might have to update the installer before continuing. If so, follow the prompts.

2. In the installer, look for the edition of Visual Studio that you installed.

For example, if you previously installed Visual Studio Community 2019 and there's an update for it, then an **Update available** message appears in the installer.

Visual Studio Installer

Installed Available

Visual Studio Community 2019 Preview
16.3.0 Preview 1.0
Free, fully-featured IDE for students, open-source and individual developers
[Release notes](#)

Visual Studio Community 2019
16.1.6
1 Update available
16.2.0
[View details](#)

Visual Studio Community 2017
15.9.14
Free, fully-featured IDE for students, open-source and individual developers
[Release notes](#)

Modify
Launch
More ▾

Developer News

Visual Studio 2019 version 16.2 Generally Available and 16.3 Preview 1
Today we are making Visual Studio 2019 version...
Thursday, July 25, 2019

Skill up on Visual Studio 2019 with Pluralsight
With the launch of Visual Studio 2019 in April, w...
Thursday, July 25, 2019

Announcing Entity Framework Core 3.0
Preview 7 and Entity Framework 6.3 Preview 7
Today, we are making new previews of EF Core...
Thursday, July 25, 2019

[View more online...](#)

Need help? Check out the [Microsoft Developer Community](#) or reach us via [Visual Studio Support](#).
Installer Version 2.3.13.627

3. Choose **Update** to install the updates.

Visual Studio Community 2019
16.1.6
1 Update available
16.2.0
[View details](#)

Update
Launch
More ▾

4. After the update is complete, you might be asked to restart your computer. If so, do so, and then start Visual Studio as you typically would.

If you aren't asked to restart your computer, choose **Launch** to start Visual Studio from the installer.

Visual Studio Community 2019
16.2.0
Free, fully-featured IDE for students, open-source and individual developers
[Release notes](#)

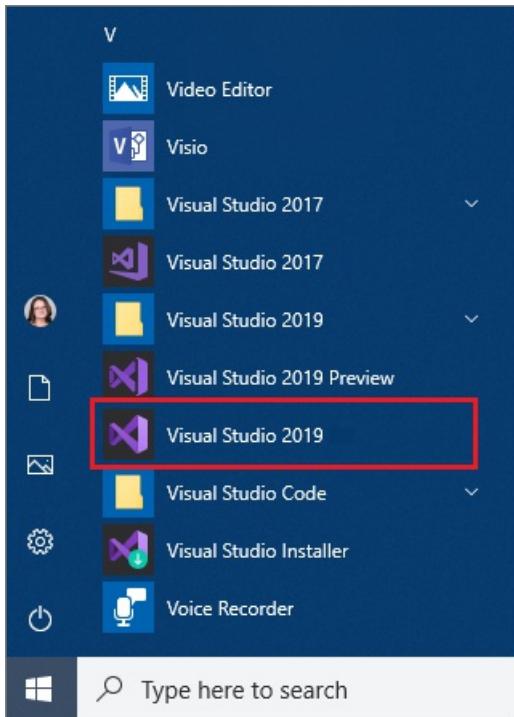
Modify
Launch
More ▾

Use the IDE

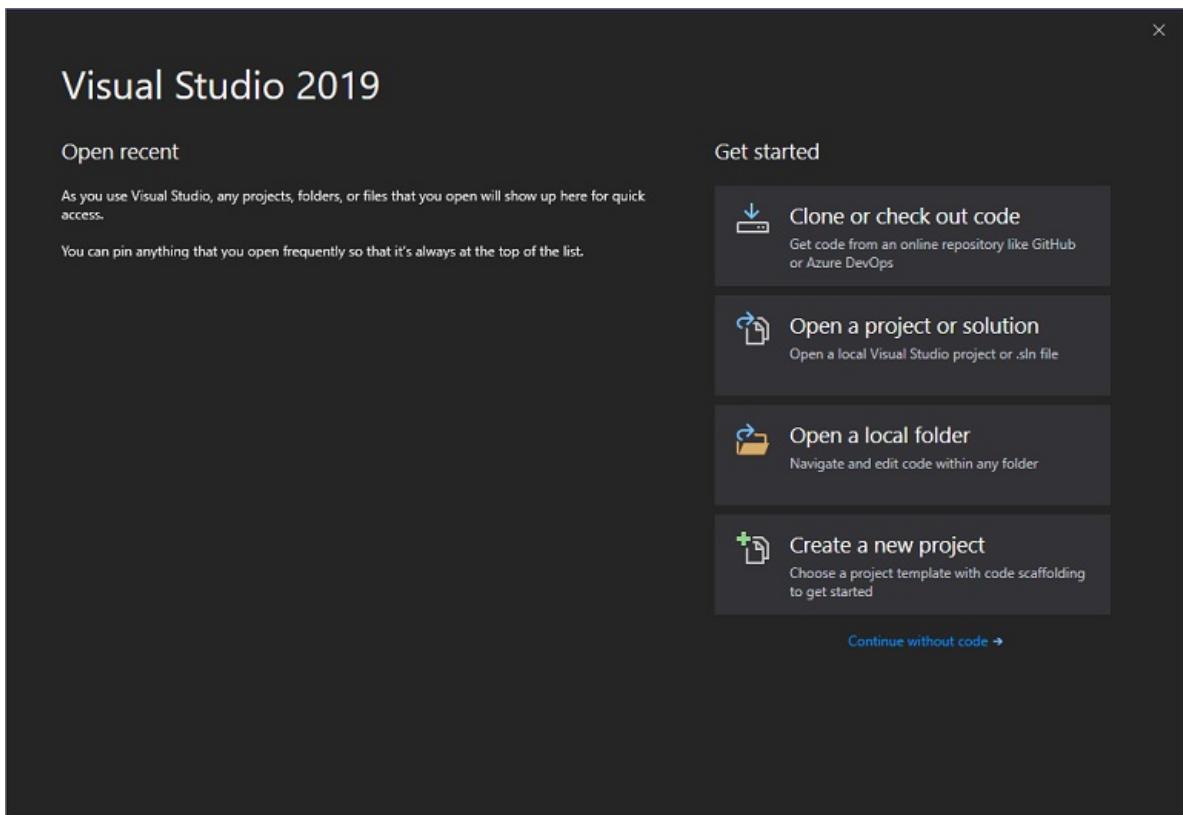
You can check for an update and then install it by using the menu bar or the search box in Visual Studio 2019.

Open Visual Studio

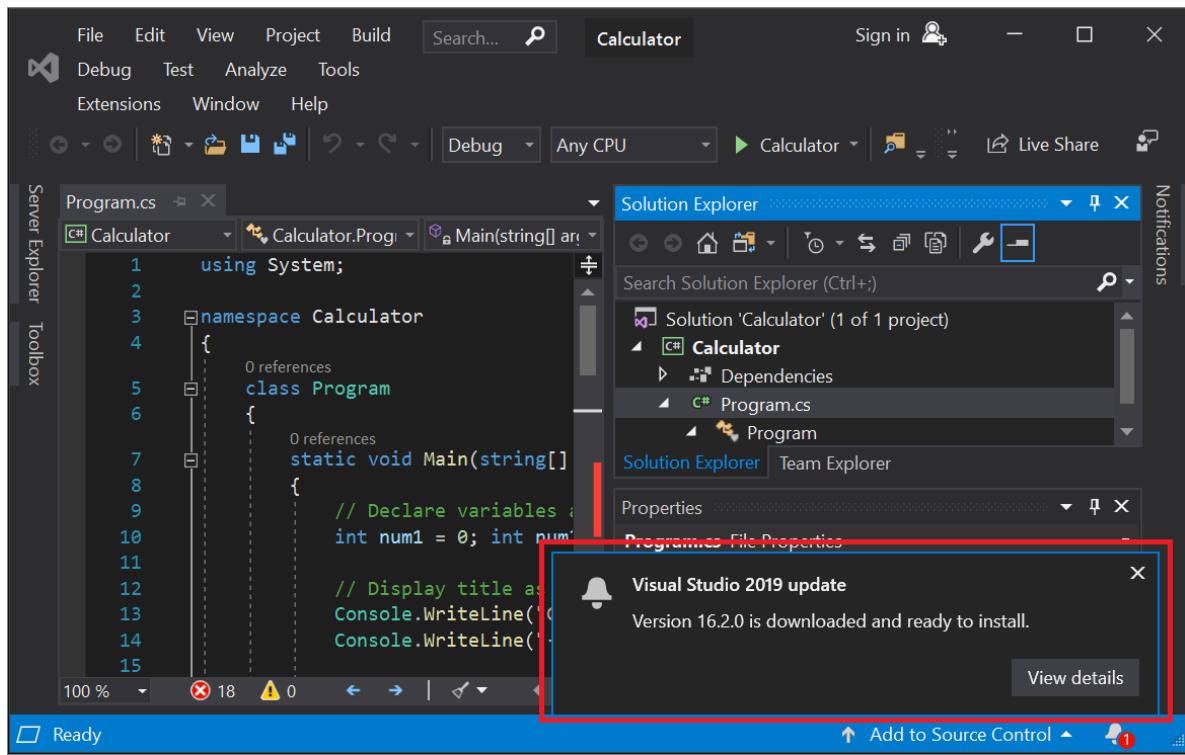
1. From the Windows **Start** menu, choose **Visual Studio 2019**.



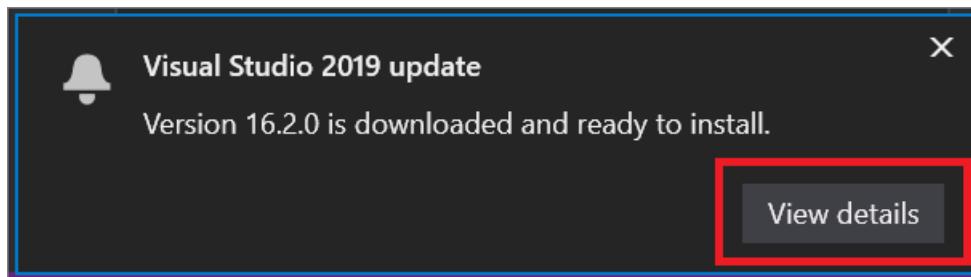
2. Under **Get started**, choose any option to open the IDE.



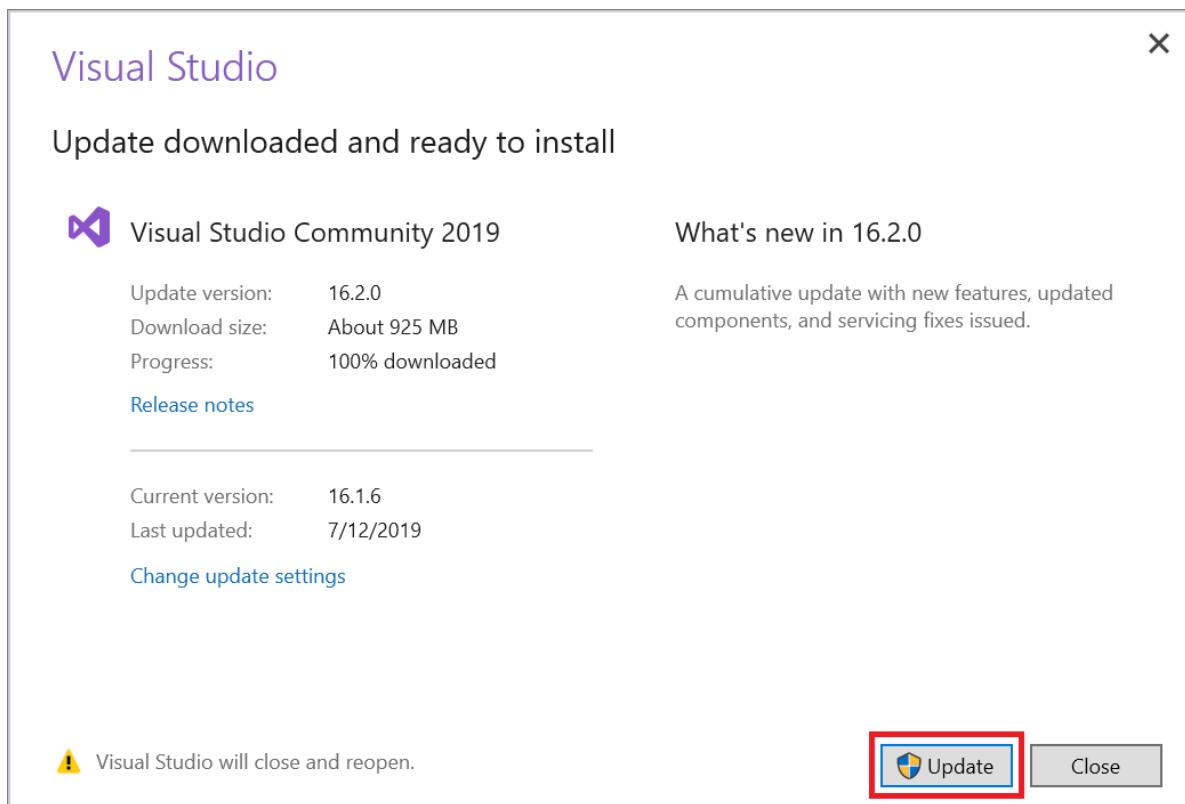
Visual Studio opens. In the IDE, a **Visual Studio 2019 update** message appears.



3. In the **Visual Studio 2019 update** message, choose **View details**.



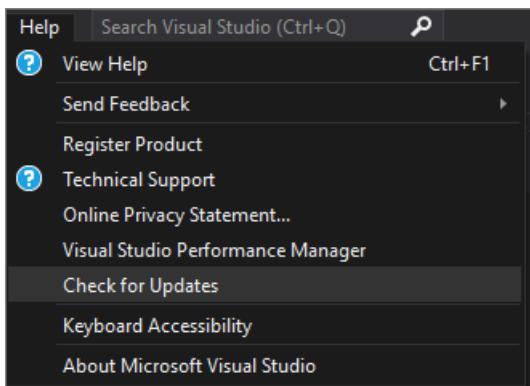
4. In the **Update downloaded and ready to install** dialog box, choose **Update**.



Visual Studio updates, closes, and then reopens.

In Visual Studio

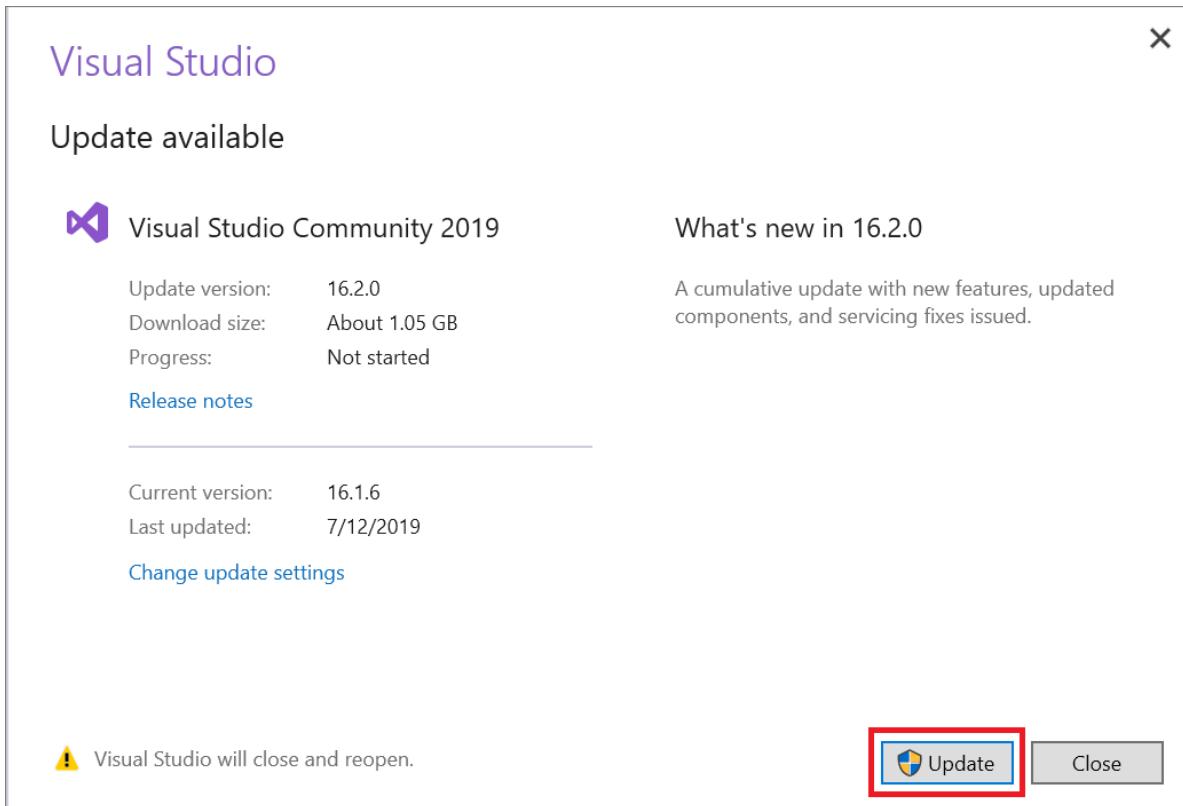
- From the menu bar, choose **Help**, and then choose **Check for Updates**.



NOTE

You can also use the search box in the IDE to check for updates. Press **Ctrl+Q**, type "check for updates", and then choose the search result that matches.

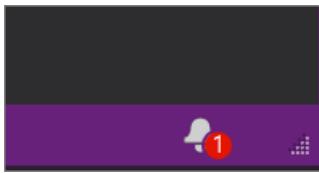
- In the **Update available** dialog box, choose **Update**.



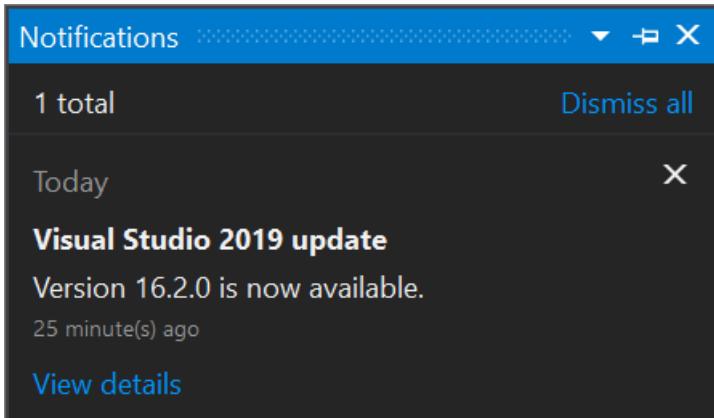
Visual Studio updates, closes, and then reopens.

Use the Notifications hub

- In Visual Studio, save your work.
- Choose the notification icon from the lower-right corner of the Visual Studio IDE to open the **Notifications** hub.



3. In the **Notifications hub**, choose the update that you want to install, and then choose **View details**.



TIP

An update for an edition of Visual Studio 2019 is cumulative, so always choose to install the one with the most recent version number.

4. In the **Update available** dialog box, choose **Update**.

Visual Studio updates, closes, and then reopens.

Customize update settings

You can customize the update settings in Visual Studio in several different ways, such as by changing the installation mode and by selecting automatic downloads.

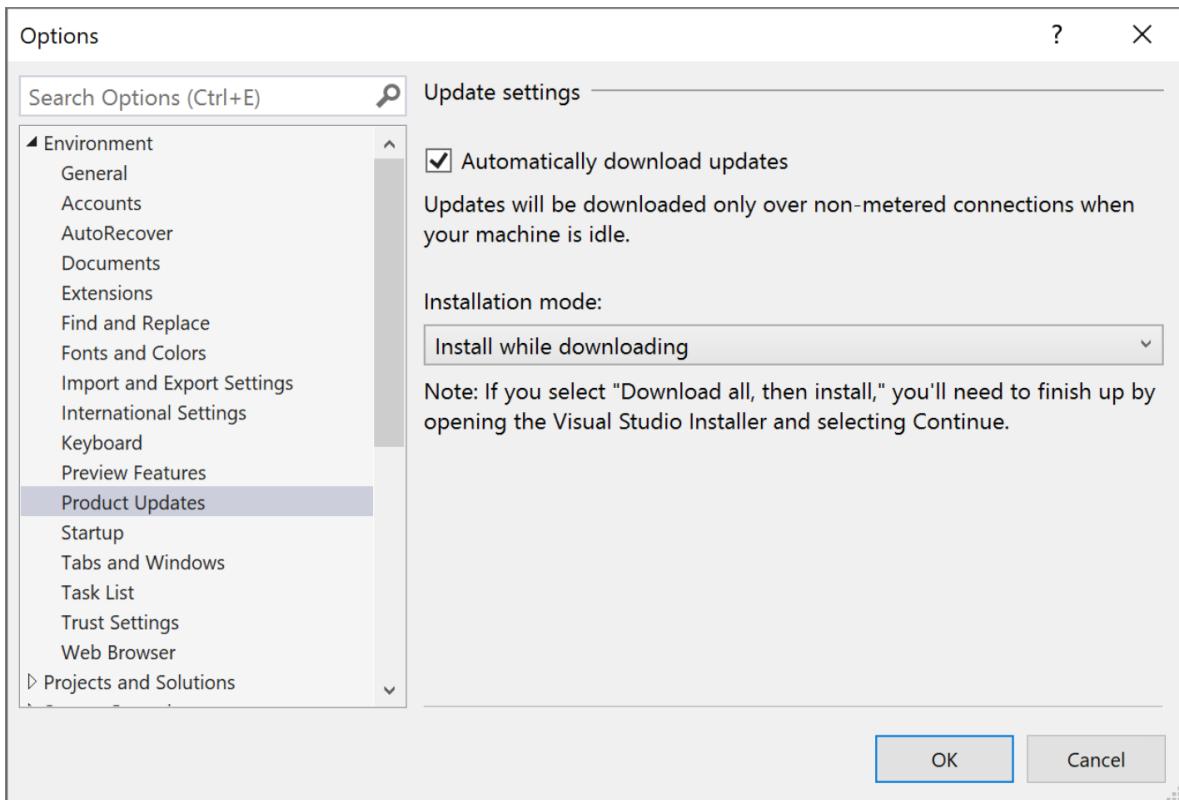
There are two installation modes to choose from:

- **Install while downloading**
- **Download all, then install**

You can also choose the **Automatically download updates** setting, which allows updates to download while your machine is idle.

Here's how:

1. On the menu bar, choose **Tools > Options**.
2. Expand **Environment**, and then choose **Product Updates**.



3. Choose the installation mode and the automatic download options you want for your Visual Studio updates.

Get support

Sometimes, things can go wrong. If your Visual Studio installation fails, see [Troubleshoot Visual Studio installation and upgrade issues](#) for step-by-step guidance.

We also offer a [live chat](#) (English only) support option for installation-related issues.

Here are a few more support options:

- Report product issues to us via the [Report a Problem](#) tool that appears both in the Visual Studio Installer and in the Visual Studio IDE.
- Suggest a feature, track product issues, and find answers in the [Visual Studio Developer Community](#).
- Use your [GitHub](#) account to talk to us and other Visual Studio developers in the [Visual Studio conversation in the Gitter community](#).

See also

- [Install Visual Studio versions side-by-side](#)
- [Update a network-based installation of Visual Studio](#)
- [Update Visual Studio while on a servicing baseline](#)
- [Control updates to network-based Visual Studio deployments](#)
- [Modify Visual Studio](#)
- [Uninstall Visual Studio](#)

Modify Visual Studio by adding or removing workloads and components

12/4/2019 • 3 minutes to read • [Edit Online](#)

It's easy to modify Visual Studio so that it includes only what you want, when you want it. To do so, open the Visual Studio Installer to add or remove workloads and components.

Not only have we made it easier for you to personalize Visual Studio to match the tasks you want to accomplish, we've also made it easier to customize Visual Studio, too. To do so, start the new Visual Studio Installer and make the changes you want.

Here's how.

IMPORTANT

To install, update, or modify Visual Studio, you must log on with an account that has administrative permissions. For more information, see [User permissions and Visual Studio](#).

Modify workloads

[Workloads](#) contain the features you need for the programming language or platform that you're using. Use workloads to modify Visual Studio so that it supports the work you want to do, when you want to do it.

Workloads contain the features you need for the programming language or platform that you're using. Use workloads to modify Visual Studio so that it supports the work you want to do, when you want to do it.

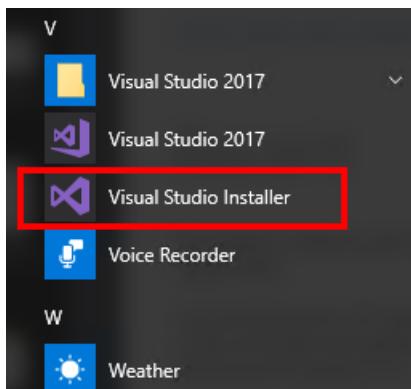
NOTE

The following procedure assumes that you have an internet connection.

For more information about how to modify a previously created [offline installation](#) of Visual Studio, see both the [Update a network-based installation of Visual Studio](#) page and the [Control updates to network-based Visual Studio deployments](#) page.

1. Find the Visual Studio Installer on your computer.

For example, on a computer running Windows 10, select **Start**, and then scroll to the letter **V**, where it's listed as **Visual Studio Installer**.



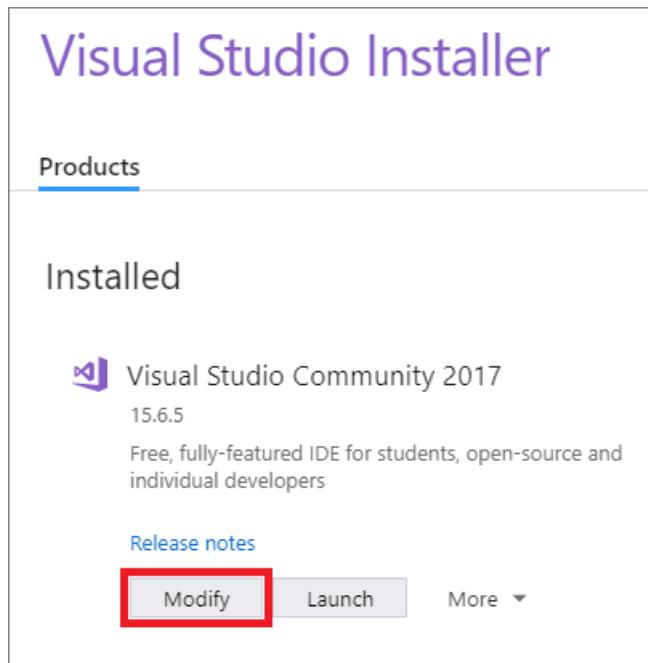
TIP

On some computers, the Visual Studio Installer might be listed under the letter "M" as the **Microsoft Visual Studio Installer**.

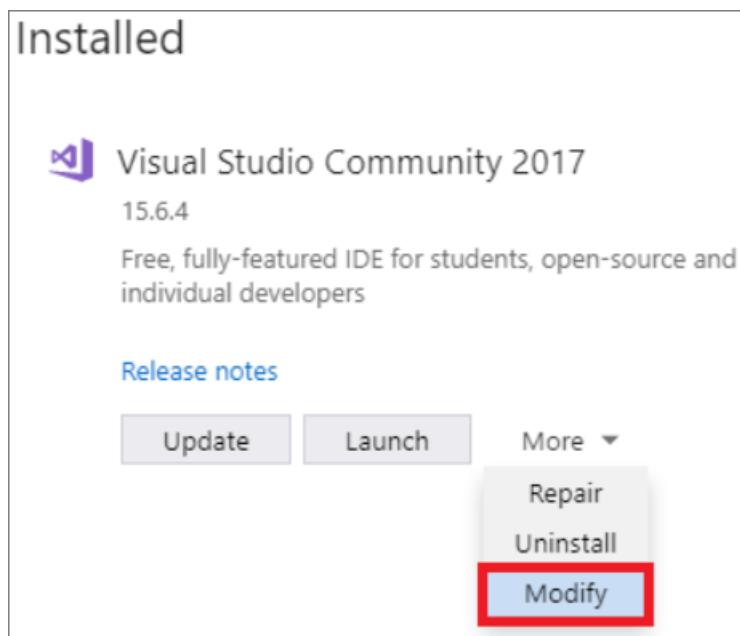
Alternatively, you can find the Visual Studio Installer in the following location:

C:\Program Files (x86)\Microsoft Visual Studio\Installer\vs_installer.exe

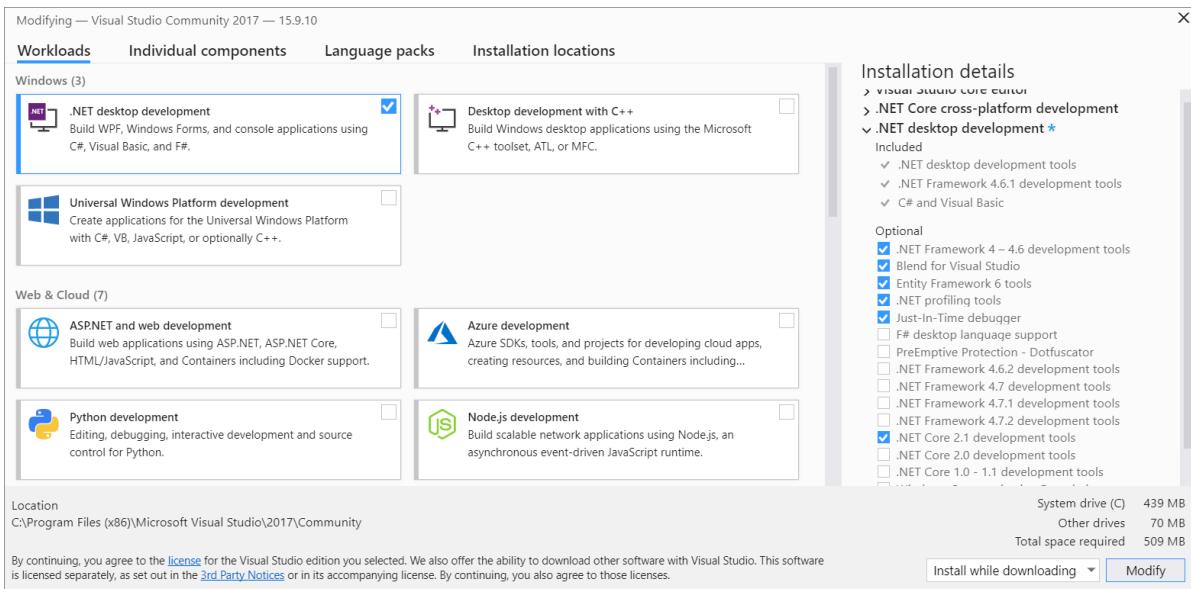
2. Click or tap to start the installer, and then choose **Modify**.



If you have an update pending, the **Modify** button is in a different place. This way, you can modify Visual Studio without updating it, should you choose to do so. Click **More**, and then choose **Modify**.



3. From the **Workloads** screen, select or deselect the workloads that you want to install or uninstall.

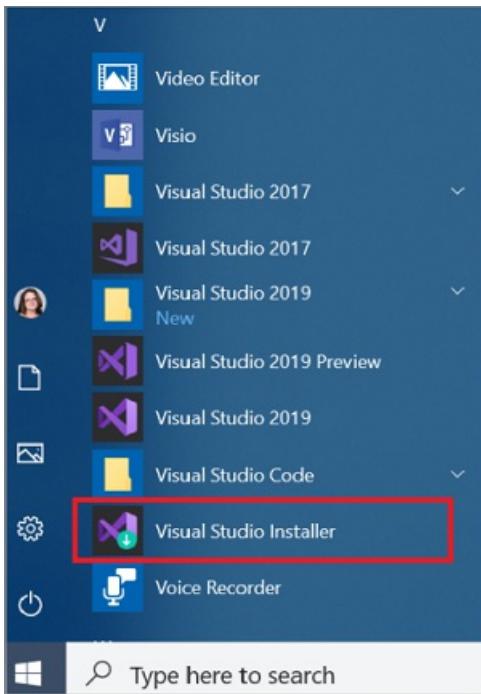


4. Choose **Modify** again.

5. After the new workloads and components are installed, choose **Launch**.

1. Find the Visual Studio Installer on your computer.

For example, on a computer running Windows 10, select **Start**, and then scroll to the letter **V**, where it's listed as **Visual Studio Installer**.



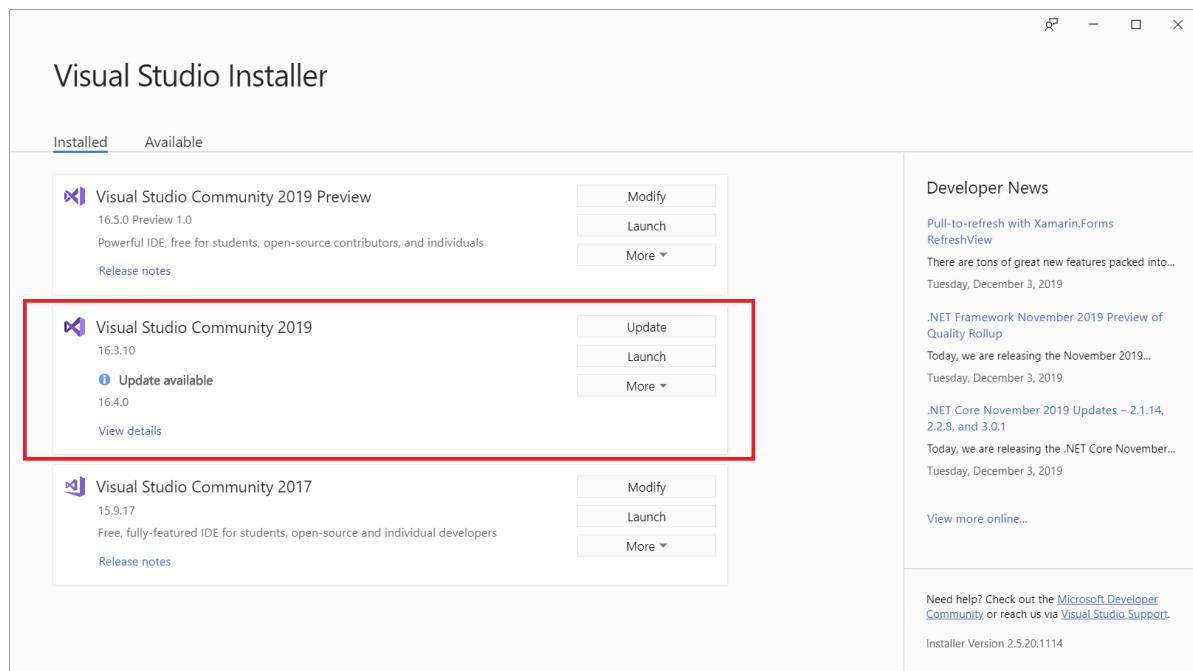
NOTE

You can also find the Visual Studio Installer in the following location:

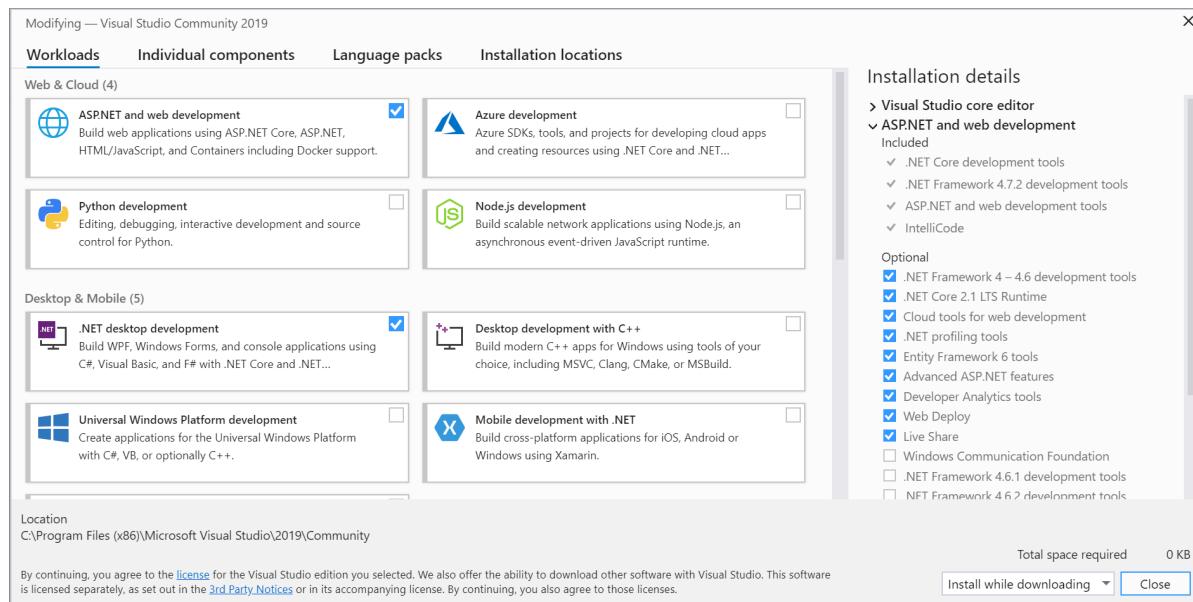
```
C:\Program Files (x86)\Microsoft Visual Studio\Installer\vs_installer.exe
```

You might have to update the installer before continuing. If so, follow the prompts.

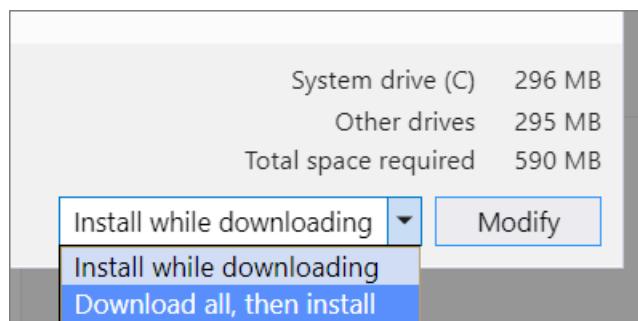
2. In the installer, look for the edition of Visual Studio that you installed, and then choose **Modify**.



3. In the **Workloads** tab, select or deselect the workloads that you want to install or uninstall.



4. Choose whether you want to accept the default **Install while downloading** option or the **Download all, then install** option.



The "Download all, then install" option is handy if you want to download first and then install later.

5. Choose **Modify**.

6. After the new workloads and components are installed, choose **Launch** from the Visual Studio Installer.

Modify individual components

If you don't want to install workloads to customize your Visual Studio installation, choose the **Individual Components** tab from the Visual Studio Installer, select what you want, and then follow the prompts.

Get support

Sometimes, things can go wrong. If your Visual Studio installation fails, see [Troubleshoot Visual Studio installation and upgrade issues](#) for step-by-step guidance.

We also offer a [live chat](#) (English only) support option for installation-related issues.

Here are a few more support options:

- Report product issues to us via the [Report a Problem](#) tool that appears both in the Visual Studio Installer and in the Visual Studio IDE.
- Suggest a feature, track product issues, and find answers in the [Visual Studio Developer Community](#).
- Use your [GitHub](#) account to talk to us and other Visual Studio developers in the [Visual Studio conversation in the Gitter community](#).

See also

- [List of Visual Studio workload & component IDs](#)
- [Update Visual Studio](#)
- [Update a network-based installation of Visual Studio](#)
- [Update Visual Studio while on a servicing baseline](#)
- [Control updates to network-based Visual Studio deployments](#)
- [Uninstall Visual Studio](#)

Repair Visual Studio

7/31/2019 • 2 minutes to read • [Edit Online](#)

Sometimes your Visual Studio installation becomes damaged or corrupted. A repair can fix this.

1. Find the **Visual Studio Installer** on your computer.

For example, on a computer running Windows 10 Anniversary Update or later, select **Start**, and then scroll to the letter **V**, where it's listed as **Visual Studio Installer**.

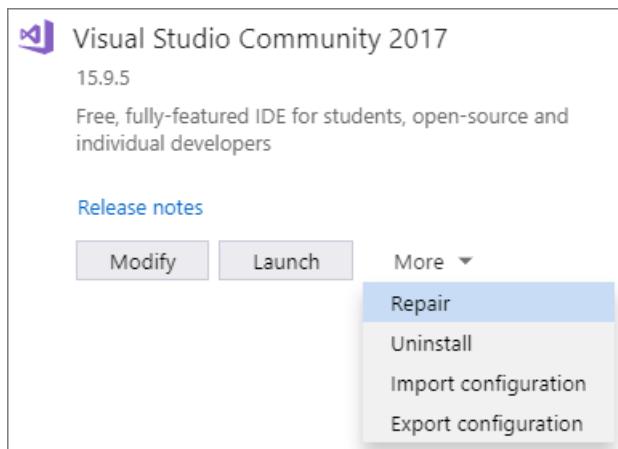
NOTE

On some computers, the Visual Studio Installer might be listed under the letter "**M**" as the **Microsoft Visual Studio Installer**.

Alternatively, you can find the Visual Studio Installer in the following location:

```
C:\Program Files (x86)\Microsoft Visual Studio\Installer\vs_installer.exe
```

2. Open the installer, choose **More**, and then choose **Repair**.



NOTE

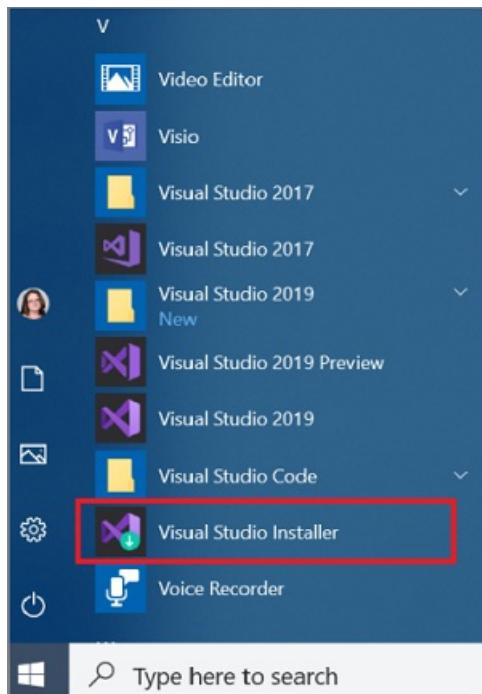
Repairing Visual Studio will reset the environment. Local customizations like per-user extensions installed without elevation, user settings, and profiles will be removed. Your synchronized settings such as themes, colors, key bindings will be restored.

TIP

The **Repair** option appears only for installed instances of Visual Studio. If you do not see the **Repair** option, chances are that you've selected **More** in a version that's listed in the Visual Studio Installer as "Available" rather than "Installed".

1. Find the Visual Studio Installer on your computer.

For example, on a computer running Windows 10, select **Start**, and then scroll to the letter **V**, where it's listed as **Visual Studio Installer**.



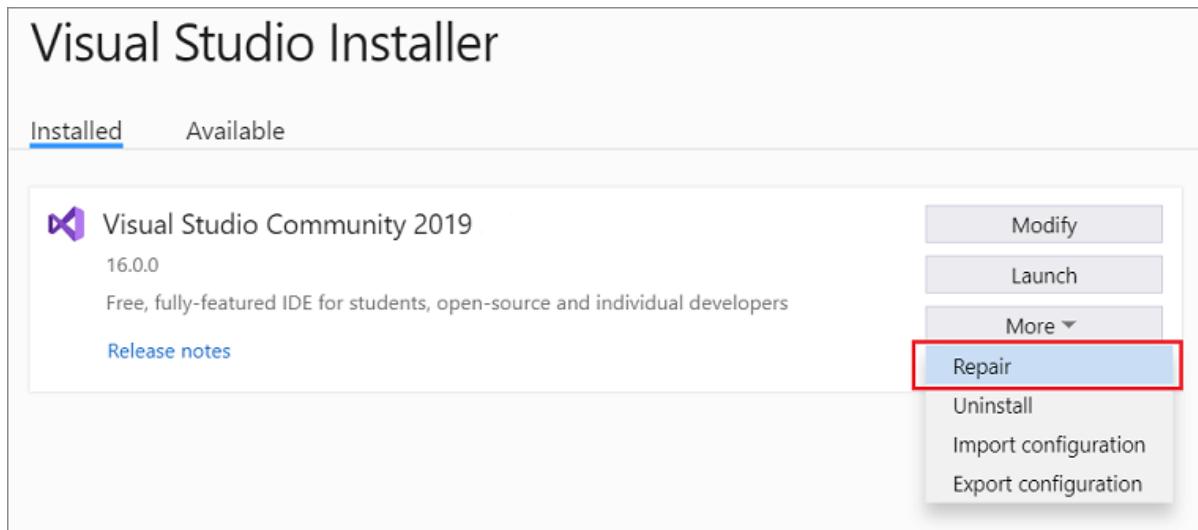
NOTE

You can also find the Visual Studio Installer in the following location:

```
C:\Program Files (x86)\Microsoft Visual Studio\Installer\vs_installer.exe
```

You might have to update the installer before continuing. If so, follow the prompts.

2. In the installer, look for the edition of Visual Studio that you installed. Next, choose **More**, and then choose **Repair**.



NOTE

Repairing Visual Studio will reset the environment. Local customizations like per-user extensions installed without elevation, user settings, and profiles will be removed. Your synchronized settings such as themes, colors, key bindings will be restored.

TIP

The **Repair** option appears only for installed instances of Visual Studio. If you do not see the **Repair** option, chances are that you've selected **More** in a version that's listed in the Visual Studio Installer as "Available" rather than "Installed".

Get support

Sometimes, things can go wrong. If your Visual Studio installation fails, see [Troubleshoot Visual Studio installation and upgrade issues](#) for step-by-step guidance.

We also offer a [live chat](#) (English only) support option for installation-related issues.

Here are a few more support options:

- Report product issues to us via the [Report a Problem](#) tool that appears both in the Visual Studio Installer and in the Visual Studio IDE.
- Suggest a feature, track product issues, and find answers in the [Visual Studio Developer Community](#).
- Use your [GitHub](#) account to talk to us and other Visual Studio developers in the [Visual Studio conversation in the Gitter community](#).

See also

- [Install Visual Studio](#)
- [Update Visual Studio](#)
- [Uninstall Visual Studio](#)
- [Troubleshooting Visual Studio installation and upgrade issues](#)

Uninstall Visual Studio

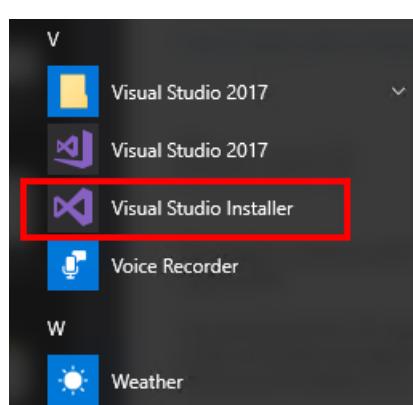
12/4/2019 • 2 minutes to read • [Edit Online](#)

This page walks you through uninstalling Visual Studio, our integrated suite of productivity tools for developers.

NOTE

This topic applies to Visual Studio on Windows. For Visual Studio for Mac, see [Uninstall Visual Studio for Mac](#).

1. Find the Visual Studio Installer on your computer.



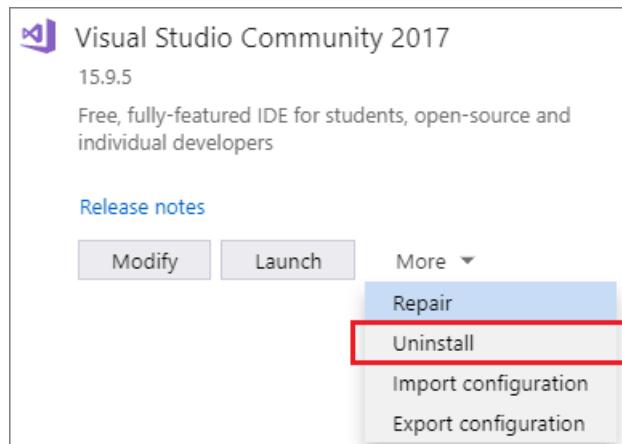
NOTE

On some computers, the Visual Studio Installer might be listed under the letter "M" as the **Microsoft Visual Studio Installer**.

Alternatively, you can find the Visual Studio Installer in the following location:

```
C:\Program Files (x86)\Microsoft Visual Studio\Installer\vs_installer.exe
```

2. In the installer, look for the edition of Visual Studio that you installed. Next, choose **More**, and then choose **Uninstall**.



3. Click **OK** to confirm your choice.

If you change your mind later and want to reinstall Visual Studio 2017, start the Visual Studio Installer again, and then select **Install** from the selection screen.

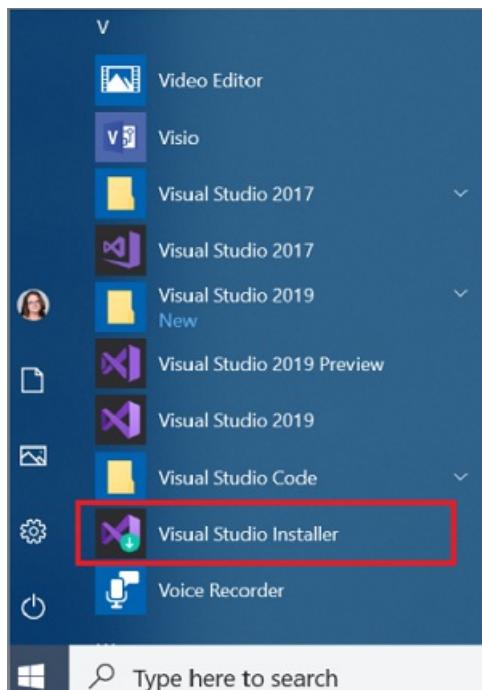
Uninstall Visual Studio Installer

To completely remove all installations of Visual Studio 2017 and the Visual Studio Installer from your machine, uninstall it from Apps & Features.

1. In Windows 10, type **Apps and Features** in the "Type here to search" box.
2. Find **Microsoft Visual Studio 2017** (or, **Visual Studio 2017**).
3. Choose **Uninstall**.
4. Then, find **Microsoft Visual Studio Installer**.
5. Choose **Uninstall**.

1. Find the Visual Studio Installer on your computer.

For example, on a computer running Windows 10, select **Start**, and then scroll to the letter **V**, where it's listed as **Visual Studio Installer**.



NOTE

You can also find the Visual Studio Installer in the following location:

```
C:\Program Files (x86)\Microsoft Visual Studio\Installer\vs_installer.exe
```

You might have to update the installer before continuing. If so, follow the prompts.

2. In the installer, look for the edition of Visual Studio that you installed. Next, choose **More**, and then choose **Uninstall**.

Visual Studio Installer

Installed Available

Visual Studio Community 2019

16.0.0

Free, fully-featured IDE for students, open-source and individual developers

[Release notes](#)

Modify

Launch

More ▾

Repair

Uninstall

Import configuration

Export configuration

- Click **OK** to confirm your choice.

Uninstall Visual Studio

You are about to uninstall the following product:

Visual Studio Community 2019

C:\Program Files (x86)\Microsoft Visual Studio\2019\Community

Click OK to continue.

[Get troubleshooting tips](#)

OK

Cancel

If you change your mind later and want to reinstall Visual Studio 2019, start the Visual Studio Installer again, choose the **Available** tab, choose the edition of Visual Studio that you want to install, and then select **Install**.

Uninstall Visual Studio Installer

To remove all installations of Visual Studio 2019 and the Visual Studio Installer from your machine, uninstall it from Apps & Features.

- In Windows 10, type **Apps and Features** in the "Type here to search" box.
- Find **Visual Studio 2019**.
- Choose **Uninstall**.
- Then, find **Microsoft Visual Studio Installer**.
- Choose **Uninstall**.

Get support

Sometimes, things can go wrong. If your Visual Studio installation fails, see [Troubleshoot Visual Studio installation and upgrade issues](#) for step-by-step guidance.

We also offer a [live chat](#) (English only) support option for installation-related issues.

Here are a few more support options:

- Report product issues to us via the [Report a Problem](#) tool that appears both in the Visual Studio Installer and

in the Visual Studio IDE.

- Suggest a feature, track product issues, and find answers in the [Visual Studio Developer Community](#).
- Use your [GitHub](#) account to talk to us and other Visual Studio developers in the [Visual Studio conversation in the Gitter community](#).

See also

- [Modify Visual Studio](#)
- [Update Visual Studio](#)
- [Uninstall Visual Studio for Mac](#)

Visual Studio administrator guide

8/26/2019 • 5 minutes to read • [Edit Online](#)

In enterprise environments, system administrators typically deploy installations to end-users from a network share or by using systems management software. We've designed the Visual Studio setup engine to support enterprise deployment by giving system administrators the ability to create a network install location, to pre-configure installation defaults, to deploy product keys during the installation process, and to manage product updates after a successful rollout.

This administrator guide provides scenario-based guidance for enterprise deployment in networked environments.

Before you begin

Before you deploy Visual Studio across your organization, there are a few decisions to make and tasks to complete:

- Make sure that each target computer meets the [minimum installation requirements](#).
- Decide on your servicing needs.

If your company needs to stay on a feature set longer but still wants to get regular servicing updates, plan to use a servicing baseline. For more information, see the **Support options for Enterprise and Professional customers** section of the [Visual Studio product lifecycle and servicing](#) page, as well as the [How to: Update Visual Studio while on a servicing baseline](#) page.

If you plan to apply servicing updates along with cumulative feature updates, then you can choose the latest bits.

- Decide on the update model.

Where do you want individual client machines to get updates? Specifically, decide whether you want to get updates from the internet or from a company-wide local share. Then, if you choose to use a local share, decide whether individual users can update their own clients or if you want an admin to update the clients programmatically.

- Decide which [workloads and components](#) your company needs.
- Decide whether to use a [response file](#) (that simplifies managing details in the script file).
- Decide if you want to enable Group Policy, and if you want to configure Visual Studio to disable customer feedback on individual computers.
- Make sure that each target computer meets the [minimum installation requirements](#).
- Decide on your servicing needs.

If your company needs to stay on a feature set longer but still wants to get regular servicing updates, plan to use a servicing baseline. For more information, see the **Support for older versions of Visual Studio** section of the [Visual Studio product lifecycle and servicing](#) page, as well as the [How to: Update Visual Studio while on a servicing baseline](#) page.

If you plan to apply servicing updates along with cumulative feature updates, then you can choose the latest bits.

- Decide on the update model.

Where do you want individual client machines to get updates? Specifically, decide whether you want to get updates from the internet or from a company-wide local share. Then, if you choose to use a local share, decide whether individual users can update their own clients or if you want an admin to update the clients programmatically.

- Decide which [workloads and components](#) your company needs.
- Decide whether to use a [response file](#) (that simplifies managing details in the script file).
- Decide if you want to enable Group Policy, and if you want to configure Visual Studio to disable customer feedback on individual computers.

Step 1 - Download Visual Studio product files

- [Select the workloads and components](#) that you want to install.
- [Create a network share for the Visual Studio product files](#).

Step 2 - Build an installation script

- Build an installation script that uses [command-line parameters](#) to control the installation.

NOTE

You can simplify scripts by using a [response file](#). Make sure to create a response file that contains your default installation option.

- (Optional) [Apply a volume license product key](#) as part of the installation script so that users don't need to activate the software separately.
- (Optional) Update the network layout to [control when and from where product updates are delivered to your end-users](#).
- (Optional) Set registry policies that affect the deployment of Visual Studio such as where some packages shared with other versions or instances are installed, [where packages are cached](#) or [whether packages are cached](#).
- (Optional) Set Group Policy. You can also [configure Visual Studio to disable customer feedback](#) on individual computers.

Step 3 - Deploy

- Use your deployment technology of choice to execute your script onto your target developer workstations.

Step 4 - Deploy updates

- [Refresh your network location with the latest updates](#) to Visual Studio by running the command you used in step 1 on a regular basis to add updated components.

You can update Visual Studio by using an update script. To do so, use the `update` command-line parameter.

Step 5 - (Optional) Use Visual Studio tools

We have several tools available to help you [detect and manage installed Visual Studio instances](#) on client

machines.

Step 1 - Download Visual Studio product files

- Select the workloads and components that you want to install.
- Create a network share for the Visual Studio product files.

Step 2 - Build an installation script

- Build an installation script that uses command-line parameters to control the installation.

NOTE

You can simplify scripts by using a response file. Make sure to create a response file that contains your default installation option.

- (Optional) Apply a volume license product key as part of the installation script so that users don't need to activate the software separately.
- (Optional) Update the network layout to control when and from where product updates are delivered to your end-users.
- (Optional) Set registry policies that affect the deployment of Visual Studio such as where some packages shared with other versions or instances are installed, where packages are cached or whether packages are cached.
- (Optional) Set Group Policy. You can also configure Visual Studio to disable customer feedback on individual computers.

Step 3 - Deploy

- Use your deployment technology of choice to execute your script onto your target developer workstations.

Step 4 - Deploy updates

- Refresh your network location with the latest updates to Visual Studio by running the command you used in step 1 on a regular basis to add updated components.

You can update Visual Studio by using an update script. To do so, use the `update` command-line parameter.

Step 5 - (Optional) Use Visual Studio tools

We have several tools available to help you detect and manage installed Visual Studio instances on client machines.

Get support

Sometimes, things can go wrong. If your Visual Studio installation fails, see Troubleshoot Visual Studio installation and upgrade issues for step-by-step guidance.

We also offer a live chat (English only) support option for installation-related issues.

Here are a few more support options:

- Report product issues to us via the Report a Problem tool that appears both in the Visual Studio Installer and

in the Visual Studio IDE.

- Suggest a feature, track product issues, and find answers in the [Visual Studio Developer Community](#).
- Use your [GitHub](#) account to talk to us and other Visual Studio developers in the [Visual Studio conversation in the Gitter community](#).

See also

- [Command-line parameter examples](#)
- [Install certificates required for Visual Studio offline installation](#)
- [Import or export installation configurations](#)
- [Visual Studio Setup Archives](#)
- [Visual Studio product lifecycle and servicing](#)
- [Synchronous autoload settings](#)

Use command-line parameters to install Visual Studio

10/24/2019 • 18 minutes to read • [Edit Online](#)

When you install Visual Studio from a command prompt, you can use a variety of command-line parameters to control or customize the installation. From the command line, you can perform the following actions:

- Start the install with certain options preselected.
- Automate the installation process.
- Create a cache (layout) of the installation files for later use.

The command-line options are used in conjunction with the setup bootstrapper, which is the small (1 MB) file that initiates the download process. The bootstrapper is the first executable that is launched when you download from the Visual Studio site.

To get a bootstrapper for Visual Studio 2017, see the [Visual Studio previous versions](#) download page for details on how to do so.

Use the following links to get a direct link to the latest release bootstrapper for the product edition that you're installing:

- [Visual Studio 2019 Enterprise](#)
- [Visual Studio 2019 Professional](#)
- [Visual Studio 2019 Community](#)

Your bootstrapper file should match or be similar to one of the following file names:

- vs_enterprise.exe
- vs_professional.exe
- vs_community.exe

TIP

If you previously downloaded a bootstrapper file and want to verify its version, here's how. In Windows, open File Explorer, right-click the bootstrapper file, choose **Properties**, choose the **Details** tab, and then view the **Product version** number. To match that number to a release of Visual Studio, see the [Visual Studio build numbers and release dates](#) page.

Command-line parameters

Visual Studio command-line parameters are case-insensitive.

Syntax: `vs_enterprise.exe [command] <options>...`

Replace `vs_enterprise.exe` as appropriate for the product edition you're installing. (Alternatively, you can use `vs_installer.exe`.)

TIP

For more examples of how to use the command line to install Visual Studio, see the [Command-line parameter examples page](#).

COMMAND	DESCRIPTION
(blank)	Installs the product.
<code>modify</code>	Modifies an installed product.
<code>update</code>	Updates an installed product.
<code>repair</code>	Repairs an installed product.
<code>uninstall</code>	Uninstalls an installed product.
<code>export</code>	New in version 15.9: Exports installation selection to an installation configuration file. Note: Can only be used with <code>vs_installer.exe</code> .

COMMAND	DESCRIPTION
(blank)	Installs the product.
<code>modify</code>	Modifies an installed product.
<code>update</code>	Updates an installed product.
<code>repair</code>	Repairs an installed product.
<code>uninstall</code>	Uninstalls an installed product.
<code>export</code>	Exports installation selection to an installation configuration file. Note: Can only be used with <code>vs_installer.exe</code> .

Install options

INSTALL OPTION	DESCRIPTION
<code>--installPath <dir></code>	The installation directory for the instance to act upon. For the <code>install</code> command, this is Optional and is where the instance will be installed. For other commands, this is Required and is where the previously installed instance was installed.

INSTALL OPTION	DESCRIPTION
<code>--addProductLang <language-locale></code>	<p>Optional: During an install or modify operation, this determines the UI language packs that are installed to the product. It can appear multiple times on the command line to add multiple language packs. If not present, the installation uses the machine locale. For more information, see the List of language locales section on this page.</p>
<code>--removeProductLang <language-locale></code>	<p>Optional: During an install or modify operation, this determines the UI language packs that are to be removed from the product. It can appear multiple times on the command line to add multiple language packs. For more information, see the List of language locales section on this page.</p>
<code>--add <one or more workload or component IDs></code>	<p>Optional: One or more workload or component IDs to add. The required components of the artifact are installed, but not the recommended or optional components. You can control additional components globally using <code>--includeRecommended</code> and/or <code>--includeOptional</code>. To include multiple workloads or components, repeat the <code>--add</code> command (for example, <code>--add Workload1 --add Workload2</code>). For finer-grained control, you can append <code>;includeRecommended</code> or <code>;includeOptional</code> to the ID (for example, <code>--add Workload1;includeRecommended</code> or <code>--add Workload2;includeRecommended;includeOptional</code>). For more information, see the Workload and component IDs page. You can repeat this option as necessary.</p>
<code>--remove <one or more workload or component IDs></code>	<p>Optional: One or more workload or component IDs to remove. For more information, see our Workload and component IDs page. You can repeat this option as necessary.</p>
<code>--in <path></code>	<p>Optional: The URI or path to a response file.</p>
<code>--all</code>	<p>Optional: Whether to install all workloads and components for a product.</p>
<code>--allWorkloads</code>	<p>Optional: Installs all workloads and components, no recommended or optional components.</p>
<code>--includeRecommended</code>	<p>Optional: Includes the recommended components for any workloads that are installed, but not the optional components. The workloads are specified either with <code>--allWorkloads</code> or <code>--add</code>.</p>
<code>--includeOptional</code>	<p>Optional: Includes the optional components for any workloads that are installed, but not the recommended components. The workloads are specified either with <code>--allWorkloads</code> or <code>--add</code>.</p>

INSTALL OPTION	DESCRIPTION
<code>--quiet, -q</code>	Optional: Don't display any user interface while performing the installation.
<code>--passive, -p</code>	Optional: Display the user interface, but don't request any interaction from the user.
<code>--norestart</code>	Optional: If present, commands with <code>--passive</code> or <code>--quiet</code> won't automatically restart the machine (if necessary). This is ignored if neither <code>--passive</code> nor <code>-quiet</code> are specified.
<code>--nickname <name></code>	Optional: This defines the nickname to assign to an installed product. The nickname can't be longer than 10 characters.
<code>--productKey</code>	Optional: This defines the product key to use for an installed product. It's composed of 25 alphanumeric characters either in the format <code>xxxxx-xxxxx-xxxxx-xxxxx-xxxxx</code> or <code>xxxxxxxxxxxxxxxxxxxxxx</code> .
<code>--help, --?, -h, -?</code>	Display an offline version of this page.
<code>--config <path></code>	Optional and New in 15.9: During an install or modify operation, this determines the workloads and components to add based on a previously saved installation configuration file. This operation is additive and it won't remove any workload or component if they aren't present in the file. Also, items that don't apply to the product won't be added. During an export operation, this determines the location to save the installation configuration file.

INSTALL OPTION	DESCRIPTION
<code>--installPath <dir></code>	The installation directory for the instance to act upon. For the install command, this is Optional and is where the instance will be installed. For other commands, this is Required and is where the previously installed instance was installed.
<code>--addProductLang <language-locale></code>	Optional: During an install or modify operation, this determines the UI language packs that are installed to the product. It can appear multiple times on the command line to add multiple language packs. If not present, the installation uses the machine locale. For more information, see the List of language locales section on this page.
<code>--removeProductLang <language-locale></code>	Optional: During an install or modify operation, this determines the UI language packs that are to be removed from the product. It can appear multiple times on the command line to add multiple language packs. For more information, see the List of language locales section on this page.

INSTALL OPTION	DESCRIPTION
<pre>--add <one or more workload or component IDs></pre>	<p>Optional: One or more workload or component IDs to add. The required components of the artifact are installed, but not the recommended or optional components. You can control additional components globally using <code>--includeRecommended</code> and/or <code>--includeOptional</code>. To include multiple workloads or components, repeat the <code>--add</code> command (for example, <code>--add Workload1 --add Workload2</code>). For finer-grained control, you can append <code>;includeRecommended</code> or <code>;includeOptional</code> to the ID (for example, <code>--add Workload1;includeRecommended</code> or <code>--add Workload2;includeRecommended;includeOptional</code>). For more information, see the Workload and component IDs page. You can repeat this option as necessary.</p>
<pre>--remove <one or more workload or component IDs></pre>	<p>Optional: One or more workload or component IDs to remove. For more information, see our Workload and component IDs page. You can repeat this option as necessary.</p>
<pre>--in <path></pre>	<p>Optional: The URI or path to a response file.</p>
<pre>--all</pre>	<p>Optional: Whether to install all workloads and components for a product.</p>
<pre>--allWorkloads</pre>	<p>Optional: Installs all workloads and components, no recommended or optional components.</p>
<pre>--includeRecommended</pre>	<p>Optional: Includes the recommended components for any workloads that are installed, but not the optional components. The workloads are specified either with <code>--allWorkloads</code> or <code>--add</code>.</p>
<pre>--includeOptional</pre>	<p>Optional: Includes the optional components for any workloads that are installed, but not the recommended components. The workloads are specified either with <code>--allWorkloads</code> or <code>--add</code>.</p>
<pre>--quiet, -q</pre>	<p>Optional: Don't display any user interface while performing the installation.</p>
<pre>--passive, -p</pre>	<p>Optional: Display the user interface, but don't request any interaction from the user.</p>
<pre>--norestart</pre>	<p>Optional: If present, commands with <code>--passive</code> or <code>--quiet</code> won't automatically restart the machine (if necessary). This is ignored if neither <code>--passive</code> nor <code>--quiet</code> are specified.</p>
<pre>--nickname <name></pre>	<p>Optional: This defines the nickname to assign to an installed product. The nickname can't be longer than 10 characters.</p>

INSTALL OPTION	DESCRIPTION
<code>--productKey</code>	<p>Optional: This defines the product key to use for an installed product. It's composed of 25 alphanumeric characters either in the format <code>xxxxx-xxxxx-xxxxx-xxxxx-xxxxx</code> or <code>xxxxxxxxxxxxxxxxxxxxxx</code>.</p>
<code>--help, --?, -h, -?</code>	Display an offline version of this page.
<code>--config <path></code>	<p>Optional: During an install or modify operation, this determines the workloads and components to add based on a previously saved installation configuration file. This operation is additive and it won't remove any workload or component if they aren't present in the file. Also, items that don't apply to the product won't be added. During an export operation, this determines the location to save the installation configuration file.</p>

IMPORTANT

When specifying multiple workloads and components, you must repeat the `--add` or `--remove` command-line switch for each item.

Layout options

LAYOUT OPTIONS	DESCRIPTION
<code>--layout <dir></code>	Specifies a directory to create an offline install cache. For more information, see Create a network-based installation of Visual Studio .
<code>--lang <one or more language-locales></code>	<p>Optional: Used with <code>--layout</code> to prepare an offline install cache with resource packages with the specified language(s). For more information, see the List of language locales section on this page.</p>
<code>--add <one or more workload or component IDs></code>	<p>Optional: One or more workload or component IDs to add. The required components of the artifact are installed, but not the recommended or optional components. You can control additional components globally using <code>--includeRecommended</code> and/or <code>--includeOptional</code>. For finer-grained control, you can append <code>;includeRecommended</code> or <code>;includeOptional</code> to the ID (for example, <code>--add Workload1;includeRecommended</code> or <code>--add Workload2;includeOptional</code>). For more information, see the Workload and component IDs page.</p> <p>Note: If <code>--add</code> is used, only the specified workloads and components and their dependencies are downloaded. If <code>--add</code> isn't specified, all workloads and components are downloaded to the layout.</p>

LAYOUT OPTIONS	DESCRIPTION
--includeRecommended	Optional: Includes the recommended components for any workloads that are installed, but not the optional components. The workloads are specified either with --allWorkloads or --add.
--includeOptional	Optional: Includes the recommended <i>and</i> optional components for any workloads being included in the layout. The workloads are specified with --add.
--keepLayoutVersion	New in 15.3, optional: Apply changes to the layout without updating the version of the layout.
--verify	New in 15.3, optional: Verify the contents of a layout. Any corrupt or missing files are listed.
--fix	New in 15.3, optional: Verify the contents of a layout. If any files are corrupt or missing, they're redownloaded. Internet access is required to fix a layout.
--clean <one or more paths to catalogs>	New in 15.3, optional: Removes old versions of components from a layout that has been updated to a newer version.
ADVANCED INSTALL OPTIONS	DESCRIPTION
--channelId <id>	Optional: The ID of the channel for the instance to be installed. This is required for the install command, and ignored for other commands if --installPath is specified.
--channelUri <uri>	Optional: The URI of the channel manifest. If updates aren't wanted, --channelUri can point to a non-existent file (for example, --channelUri C:\doesntExist.chman). This can be used for the install command; it's ignored for other commands.
--installChannelUri <uri>	Optional: The URI of the channel manifest to use for the installation. The URI specified by --channelUri (which must be specified when --installChannelUri is specified) is used to detect updates. This can be used for the install command; it's ignored for other commands.
--installCatalogUri <uri>	Optional: The URI of the catalog manifest to use for the installation. If specified, the channel manager attempts to download the catalog manifest from this URI before using the URI in the install channel manifest. This parameter is used to support offline install, where the layout cache will be created with the product catalog already downloaded. This can be used for the install command; it's ignored for other commands.
--productId <id>	Optional The ID of the product for the instance that will be installed. This is pre-populated in normal installation conditions.

ADVANCED INSTALL OPTIONS	DESCRIPTION
<code>--wait</code>	<p>Optional: The process will wait until the install is completed before returning an exit code. This is useful when automating installations where one needs to wait for the install to finish to handle the return code from that install.</p>
<code>--locale <language-locale></code>	<p>Optional: Change the display language of the user interface for the installer itself. Setting will be persisted. For more information, see the List of language locales section on this page.</p>
<code>--cache</code>	<p>New in 15.2, optional: If present, packages will be kept after being installed for subsequent repairs. This overrides the global policy setting to be used for subsequent installs, repairs, or modifications. The default policy is to cache packages. This is ignored for the uninstall command. Read how to disable or move the package cache for more information.</p>
<code>--nocache</code>	<p>New in 15.2, optional: If present, packages will be deleted after being installed or repaired. They'll be downloaded again only if needed and deleted again after use. This overrides the global policy setting to be used for subsequent installs, repairs, or modifications. The default policy is to cache packages. This is ignored for the uninstall command. Read how to disable or move the package cache for more information.</p>
<code>--noUpdateInstaller</code>	<p>New in 15.2, optional: If present, prevents the installer from updating itself when quiet is specified. The installer will fail the command and return a non-zero exit code if noUpdateInstaller is specified with quiet when an installer update is required.</p>
<code>--noweb</code>	<p>New in 15.3, optional: If present, Visual Studio setup uses the files in your layout directory to install Visual Studio. If a user tries to install components that aren't in the layout, setup fails. For more information, see Deploying from a network installation.</p> <p>Important: This switch doesn't stop Visual Studio setup from checking for updates. For more information, see Control updates to network-based Visual Studio deployments.</p>
<code>--path <name>=<path></code>	<p>New in 15.7, optional: Used to specify custom install paths for the installation. Supported path names are shared, cache, and install.</p>
<code>--path cache=<path></code>	<p>New in 15.7, optional: Uses the location you specify to download installation files. This location can only be set the first time that Visual Studio is installed. Example:</p> <pre data-bbox="874 1922 1168 1956"><code>--path cache="C:\VS\cache"</code></pre>

ADVANCED INSTALL OPTIONS	DESCRIPTION
<pre>--path shared=<path></pre>	<p>New in 15.7, optional: Contains shared files for side-by-side Visual Studio installations. Some tools and SDKs install to a location on this drive, while some others might override this setting and install to another drive.</p> <p>Example: <code>--path shared="C:\VS\shared"</code></p> <p>Important: This can be set only once and on the first time that Visual Studio is installed.</p>
LAYOUT OPTIONS	DESCRIPTION
<pre>--layout <dir></pre>	<p>Specifies a directory to create an offline install cache. For more information, see Create a network-based installation of Visual Studio.</p>
<pre>--lang <one or more language-locales></pre>	<p>Optional: Used with <code>--layout</code> to prepare an offline install cache with resource packages with the specified language(s). For more information, see the List of language locales section on this page.</p>
<pre>--add <one or more workload or component IDs></pre>	<p>Optional: One or more workload or component IDs to add. The required components of the artifact are installed, but not the recommended or optional components. You can control additional components globally using <code>--includeRecommended</code> and/or <code>--includeOptional</code>. For finer-grained control, you can append <code>;includeRecommended</code> or <code>;includeOptional</code> to the ID (for example, <code>--add Workload1;includeRecommended</code> or <code>--add Workload2;includeOptional</code>). For more information, see the Workload and component IDs page.</p> <p>Note: If <code>--add</code> is used, only the specified workloads and components and their dependencies are downloaded. If <code>--add</code> isn't specified, all workloads and components are downloaded to the layout.</p>
<pre>--includeRecommended</pre>	<p>Optional: Includes the recommended components for any workloads that are installed, but not the optional components. The workloads are specified either with <code>--allWorkloads</code> or <code>--add</code>.</p>
<pre>--includeOptional</pre>	<p>Optional: Includes the recommended <i>and</i> optional components for any workloads being included in the layout. The workloads are specified with <code>--add</code>.</p>
<pre>--keepLayoutVersion</pre>	<p>Optional: Apply changes to the layout without updating the version of the layout.</p>
<pre>--verify</pre>	<p>Optional: Verify the contents of a layout. Any corrupt or missing files are listed.</p>

LAYOUT OPTIONS	DESCRIPTION
--fix	Optional: Verify the contents of a layout. If any files are corrupt or missing, they're redownloaded. Internet access is required to fix a layout.
--clean <one or more paths to catalogs>	Optional: Removes old versions of components from a layout that has been updated to a newer version.
ADVANCED INSTALL OPTIONS	DESCRIPTION
--channelId <id>	Optional: The ID of the channel for the instance to be installed. This is required for the install command, and ignored for other commands if --installPath is specified.
--channelUri <uri>	Optional: The URI of the channel manifest. If updates aren't wanted, --channelUri can point to a non-existent file (for example, --channelUri C:\doesntExist.chman). This can be used for the install command; it's ignored for other commands.
--installChannelUri <uri>	Optional: The URI of the channel manifest to use for the installation. The URI specified by --channelUri (which must be specified when --installChannelUri is specified) is used to detect updates. This can be used for the install command; it's ignored for other commands.
--installCatalogUri <uri>	Optional: The URI of the catalog manifest to use for the installation. If specified, the channel manager attempts to download the catalog manifest from this URI before using the URI in the install channel manifest. This parameter is used to support offline install, where the layout cache will be created with the product catalog already downloaded. This can be used for the install command; it's ignored for other commands.
--productId <id>	Optional The ID of the product for the instance that will be installed. This is pre-populated in normal installation conditions.
--wait	Optional: The process will wait until the install is completed before returning an exit code. This is useful when automating installations where one needs to wait for the install to finish to handle the return code from that install.
--locale <language-locale>	Optional: Change the display language of the user interface for the installer itself. Setting will be persisted. For more information, see the List of language locales section on this page.
--cache	Optional: If present, packages will be kept after being installed for subsequent repairs. This overrides the global policy setting to be used for subsequent installs, repairs, or modifications. The default policy is to cache packages. This is ignored for the uninstall command. Read how to disable or move the package cache for more information.

ADVANCED INSTALL OPTIONS	DESCRIPTION
<code>--nocache</code>	<p>Optional: If present, packages will be deleted after being installed or repaired. They'll be downloaded again only if needed and deleted again after use. This overrides the global policy setting to be used for subsequent installs, repairs, or modifications. The default policy is to cache packages. This is ignored for the uninstall command. Read how to disable or move the package cache for more information.</p>
<code>--noUpdateInstaller</code>	<p>Optional: If present, prevents the installer from updating itself when quiet is specified. The installer will fail the command and return a non-zero exit code if noUpdateInstaller is specified with quiet when an installer update is required.</p>
<code>--noWeb</code>	<p>Optional: If present, Visual Studio setup uses the files in your layout directory to install Visual Studio. If a user tries to install components that aren't in the layout, setup fails. For more information, see Deploying from a network installation.</p> <p>Important: This switch doesn't stop Visual Studio setup from checking for updates. For more information, see Control updates to network-based Visual Studio deployments. New in 16.3.5: This switch prevents errors and improves performance with offline installs and updates.</p>
<code>--path <name>=<path></code>	<p>Optional: Used to specify custom install paths for the installation. Supported path names are shared, cache, and install.</p>
<code>--path cache=<path></code>	<p>Optional: Uses the location you specify to download installation files. This location can only be set the first time that Visual Studio is installed. Example: <code>--path cache="C:\VS\cache"</code></p>
<code>--path shared=<path></code>	<p>Optional: Contains shared files for side-by-side Visual Studio installations. Some tools and SDKs install to a location on this drive, while some others might override this setting and install to another drive. Example: <code>--path shared="C:\VS\shared"</code></p> <p>Important: This can be set only once and on the first time that Visual Studio is installed.</p>
<code>--path install=<path></code>	<p>Optional: Equivalent to <code>--installPath</code>. Specifically, <code>--installPath "C:\VS"</code> and <code>--path install="C:\VS"</code> are equivalent. Only one of these commands can be used at a time.</p>

List of workload IDs and component IDs

For a list of workload and component IDs sorted by Visual Studio product, see the [Visual Studio workload and component IDs](#) page.

List of language locales

LANGUAGE-LOCALE	LANGUAGE
Cs-cz	Czech
De-de	German
En-us	English
Es-es	Spanish
Fr-fr	French
It-it	Italian
Ja-jp	Japanese
Ko-kr	Korean
Pl-pl	Polish
Pt-br	Portuguese - Brazil
Ru-ru	Russian
Tr-tr	Turkish
Zh-cn	Chinese - Simplified
Zh-tw	Chinese - Traditional

Error codes

Depending on the result of the operation, the `%ERRORLEVEL%` environment variable is set to one of the following values:

VALUE	RESULT
0	Operation completed successfully
1602	Operation was canceled
1641	Operation completed successfully, and reboot was initiated
3010	Operation completed successfully, but install requires reboot before it can be used
5003	Bootstrapper failed to download installer
5004	Operation was canceled

VALUE	RESULT
5005	Bootstrapper command-line parse error
5007	Operation was blocked - the computer does not meet the requirements
-1073741510	Microsoft Visual Studio Installer was terminated (by the user or external process)
Other (for example: -1, 1, 1603)	Failure condition occurred - check the logs for more information

Each operation generates several log files in the `%TEMP%` directory that indicate the progress of the installation. Sort the folder by date and look for files that begin with `dd_bootstrapper`, `dd_client`, and `dd_setup` for the bootstrapper, the installer app, and the setup engine, respectively.

Get support

Sometimes, things can go wrong. If your Visual Studio installation fails, see [Troubleshoot Visual Studio installation and upgrade issues](#) for step-by-step guidance.

We also offer a [live chat](#) (English only) support option for installation-related issues.

Here are a few more support options:

- Report product issues to us via the [Report a Problem](#) tool that appears both in the Visual Studio Installer and in the Visual Studio IDE.
- Suggest a feature, track product issues, and find answers in the [Visual Studio Developer Community](#).
- Use your [GitHub](#) account to talk to us and other Visual Studio developers in the [Visual Studio conversation in the Gitter community](#).

See also

- [Command-line parameter examples for Visual Studio installation](#)
- [Create an offline installation of Visual Studio](#)
- [Automate Visual Studio installation with a response file](#)
- [Visual Studio workload and component IDs](#)

Command-line parameter examples for Visual Studio installation

7/5/2019 • 4 minutes to read • [Edit Online](#)

To illustrate how to [use command-line parameters to install Visual Studio](#), here are several examples that you can customize to match your needs.

In each example, `vs_enterprise.exe`, `vs_professional.exe` and `vs_community.exe` represent the respective edition of the Visual Studio bootstrapper, which is the small (approximately 1MB) file that initiates the download process. If you are using a different edition, substitute the appropriate bootstrapper name.

NOTE

All commands require administrative elevation, and a User Account Control prompt will be displayed if the process is not started from an elevated prompt.

NOTE

You can use the `^` character at the end of a command line to concatenate multiple lines into a single command. Alternatively, you can simply place these lines together onto a single row. In PowerShell, the equivalent is the backtick (`\``) character.

For lists of the workloads and components that you can install by using the command line, see the [Visual Studio workload and component IDs](#) page.

Using --installPath

- Install a minimal instance of Visual Studio, with no interactive prompts but progress displayed:

```
vs_enterprise.exe --installPath C:\minVS ^
--add Microsoft.VisualStudio.Workload.CoreEditor ^
--passive --norestart
```

- Update a Visual Studio instance by using the command line, with no interactive prompts but progress displayed:

```
vs_enterprise.exe --update --quiet --wait
vs_enterprise.exe update --wait --passive --norestart --installPath "C:\installPathVS"
```

NOTE

Both commands are required. The first command updates the Visual Studio Installer. The second command updates the Visual Studio instance. To avoid a User Account Control dialog, run the command prompt as an Administrator.

- Install a desktop instance of Visual Studio silently, with the French language pack, returning only when the product is installed.

```
vs_enterprise.exe --installPath C:\desktopVS ^
--addProductLang fr-FR ^
--add Microsoft.VisualStudio.Workload.ManagedDesktop ^
--includeRecommended --quiet --wait
```

Using --wait

- Use in batch files or scripts to wait for the Visual Studio installer to complete before the next command is executed. For batch files, an `%ERRORLEVEL%` environment variable will contain the return value of the command, as documented in the [Use command-line parameters to install Visual Studio](#) page. Some command utilities require additional parameters to wait for completion and to get the installer's return value. The following is an example of the additional parameters used with the PowerShell script command 'Start-Process':

```
start /wait vs_professional.exe --installPath "C:\VS" --passive --wait > nul
echo %errorlevel%
```

```
$exitCode = Start-Process -FilePath vs_enterprise.exe -ArgumentList "--installPath", "C:\VS", "--passive", "--wait" -Wait -PassThru
```

or

```
$startInfo = New-Object System.Diagnostics.ProcessStartInfo
$startInfo.FileName = "vs_enterprise.exe"
$startInfo.Arguments = "--all --quiet --wait"
$process = New-Object System.Diagnostics.Process
$process.StartInfo = $startInfo
$process.Start()
$process.WaitForExit()
```

- The first '--wait' is used by the Visual Studio Installer, and the second '-Wait' is used by 'Start-Process' to wait for completion. The '-PassThru' parameter is used by 'Start-Process' to use the installer's exit code for its return value.

Using --layout

- Download the Visual Studio core editor (the most minimal Visual Studio configuration). Only include the English language pack:

```
vs_community.exe --layout C:\VS
--lang en-US ^
--add Microsoft.VisualStudio.Workload.CoreEditor
```

- Download the .NET desktop and .NET web workloads along with all recommended components and the GitHub extension. Only include the English language pack:

```
vs_community.exe --layout C:\VS ^
--lang en-US ^
--add Microsoft.VisualStudio.Workload.NetWeb ^
--add Microsoft.VisualStudio.Workload.ManagedDesktop ^
--add Component.GitHub.VisualStudio ^
--includeRecommended
```

Using --all

- Start an interactive installation of all workloads and components that are available in the Visual Studio Enterprise edition:

```
vs_enterprise.exe --all
```

Using --includeRecommended

- Install a second, named instance of Visual Studio Professional on a machine with Visual Studio Community edition already installed, with support for Node.js development:

```
vs_professional.exe --installPath C:\VSforNode ^
--add Microsoft.VisualStudio.Workload.Node --includeRecommended --nickname VSforNode
```

Using --remove

- Remove the Profiling Tools component from the default installed Visual Studio instance:

```
vs_enterprise.exe modify ^
--installPath "C:\Program Files (x86)\Microsoft Visual Studio\2017\Enterprise" ^
--remove Microsoft.VisualStudio.Component.DiagnosticTools ^
--passive
```

- Remove the Profiling Tools component from the default installed Visual Studio instance:

```
vs_enterprise.exe modify ^
--installPath "C:\Program Files (x86)\Microsoft Visual Studio\2019\Enterprise" ^
--remove Microsoft.VisualStudio.Component.DiagnosticTools ^
--passive
```

Using --path

These command-line parameters are **new in 15.7**. For more information about them, see the [Use command-line parameters to install Visual Studio](#) page.

- Using the install, cache, and shared paths:

```
vs_enterprise.exe --add Microsoft.VisualStudio.Workload.CoreEditor --path install="C:\VS" --path
cache="C:\VS\cache" --path shared="C:\VS\shared"
```

- Using only the install and cache paths:

```
vs_enterprise.exe --add Microsoft.VisualStudio.Workload.CoreEditor --path install="C:\VS" --path
cache="C:\VS\cache"
```

- Using only the install and shared paths:

```
vs_enterprise.exe --add Microsoft.VisualStudio.Workload.CoreEditor --path install="C:\VS" --path
shared="C:\VS\shared"
```

- Using only the install path:

```
vs_enterprise.exe --add Microsoft.VisualStudio.Workload.CoreEditor --path install="C:\VS"
```

Using export

This command-line command is **new in 15.9**. For more information about it, see the [Use command-line parameters to install Visual Studio](#) page.

- Using export to save the selection from an installation:

```
"C:\Program Files (x86)\Microsoft Visual Studio\Installer\vs_installer.exe" export --installPath "C:\VS" --config "C:\.vsconfig"
```

- Using export to save custom selection from scratch:

```
"C:\Program Files (x86)\Microsoft Visual Studio\Installer\vs_installer.exe" export --add Microsoft.VisualStudio.Workload.ManagedDesktop --includeRecommended --config "C:\.vsconfig"
```

Using --config

This command-line parameter is **new in 15.9**. For more information about it, see the [Use command-line parameters to install Visual Studio](#) page.

- Using --config to install the workloads and components from a previously saved installation configuration file:

```
vs_enterprise.exe --config "C:\.vsconfig" --installPath "C:\VS"
```

- Using --config to add workloads and components to an existing installation:

```
vs_enterprise.exe modify --installPath "C:\VS" --config "C:\.vsconfig"
```

Get support

Sometimes, things can go wrong. If your Visual Studio installation fails, see [Troubleshoot Visual Studio installation and upgrade issues](#) for step-by-step guidance.

We also offer a [live chat](#) (English only) support option for installation-related issues.

Here are a few more support options:

- Report product issues to us via the [Report a Problem](#) tool that appears both in the Visual Studio Installer and in the Visual Studio IDE.
- Suggest a feature, track product issues, and find answers in the [Visual Studio Developer Community](#).
- Use your [GitHub](#) account to talk to us and other Visual Studio developers in the [Visual Studio conversation in the Gitter community](#).

See also

- [Visual Studio Administrator Guide](#)
- [Use command-line parameters to install Visual Studio](#)
- [Create an offline installation of Visual Studio](#)
- [Visual Studio workload and component IDs](#)

Create a network installation of Visual Studio

10/31/2019 • 9 minutes to read • [Edit Online](#)

Typically, an enterprise administrator creates a network install point to deploy to client workstations. We've designed Visual Studio to enable you to cache the files for the initial installation along with all product updates to a single folder. (This process is also referred to as *creating a layout*.)

We've done this so that client workstations can use the same network location to manage their installation even if they haven't yet updated to the latest servicing update.

NOTE

If you have multiple editions of Visual Studio in use within your enterprise (for example, both Visual Studio Professional and Visual Studio Enterprise), you must create a separate network install share for each edition.

Download the Visual Studio bootstrapper

Download a bootstrapper file for the edition of Visual Studio you want. Make sure to choose **Save**, and then choose **Open folder**.

To get a bootstrapper for Visual Studio 2017, see the [Visual Studio previous versions](#) download page for details on how to do so.

Your setup executable—or to be more specific, the bootstrapper file—should match or be similar to one of the following.

EDITION	FILENAME
Visual Studio Enterprise	vs_enterprise.exe
Visual Studio Professional	vs_professional.exe
Visual Studio Build Tools	vs_buildtools.exe

Other supported bootstrappers include **vs_feedbackclient.exe**, **vs_teamexplorer.exe**, **vs_testagent.exe**, **vs_testcontroller.exe**, and **vs_testprofessional.exe**.

Your setup executable—or to be more specific, a bootstrapper file—should match or be similar to one of the following.

EDITION	DOWNLOAD
Visual Studio Enterprise	vs_enterprise.exe
Visual Studio Professional	vs_professional.exe
Visual Studio Build Tools	vs_buildtools.exe

Other supported bootstrappers include [vs_teamexplorer.exe](#), [vs_testagent.exe](#), and [vs_testcontroller.exe](#).

TIP

If you previously downloaded a bootstrapper file and want to verify its version, here's how. In Windows, open File Explorer, right-click the bootstrapper file, choose **Properties**, choose the **Details** tab, and then view the **Product version** number. To match that number to a release of Visual Studio, see the [Visual Studio build numbers and release dates](#) page.

Create an offline installation folder

You must have an internet connection to complete this step. To create an offline installation with all languages and all features, use a command that is similar to one of the following examples.

IMPORTANT

A complete Visual Studio layout requires a minimum of 35 GB of disk space and can take some time to download. See the [Customize the network layout](#) section for details on how to create a layout with only the components you want to install.

TIP

Make sure that you run the command from your Download directory. Typically, that's `C:\Users\<username>\Downloads` on a computer running Windows 10.

- For Visual Studio Enterprise, run:

```
vs_enterprise.exe --layout c:\VSLayout
```

- For Visual Studio Professional, run:

```
vs_professional.exe --layout c:\VSLayout
```

Modify the response.json file

You can modify the `response.json` to set default values that are used when setup is run. For example, you can configure the `response.json` file to select a specific set of workloads selected automatically. See [Automate Visual Studio installation with a response file](#) for details.

And, if you run into a problem with the Visual Studio bootstrapper throwing an error when you pair it with a `response.json` file, see the "Failed to parse ID from parent process" section of the [Troubleshoot network-related errors when you install or use Visual Studio](#) page for more information on what to do.

Copy the layout to a network share

Host the layout on a network share so it can be run from other machines.

The following example uses `xcopy`. You can also use `robocopy`, should you wish.

Example:

```
xcopy /e c:\VSLayout \\server\products\VS2017
```

```
xcopy /e c:\VSLayout \\server\products\VS2019
```

IMPORTANT

To prevent an error, make sure that your full layout path is less than 80 characters.

Customize the network layout

There are several options you can use to customize your network layout. You can create a partial layout that only contains a specific set of [language locales](#), [workloads](#), [components](#), and their recommended or optional dependencies. This might be useful if you know that you're going to deploy only a subset of workloads to client workstations. Typical command-line parameters for customizing the layout include:

- `--add` to specify [workload or component IDs](#).
If `--add` is used, only those workloads and components specified with `--add` are downloaded. If `--add` isn't used, all workload and components are downloaded.
- `--includeRecommended` to include all the recommended components for the specified workload IDs
- `--includeOptional` to include all the recommended and optional components for the specified workload IDs.
- `--lang` to specify [language locales](#).

Here are a few examples of how to create a custom partial layout.

- To download all workloads and components for only one language, run:

```
vs_enterprise.exe --layout C:\VSLayout --lang en-US
```

- To download all workloads and components for multiple languages, run:

```
vs_enterprise.exe --layout C:\VSLayout --lang en-US de-DE ja-JP
```

- To download one workload for all languages, run:

```
vs_enterprise.exe --layout C:\VSLayout --add Microsoft.VisualStudio.Workload.Azure --  
includeRecommended
```

- To download two workloads and one optional component for three languages, run:

```
vs_enterprise.exe --layout C:\VSLayout --add Microsoft.VisualStudio.Workload.Azure --add  
Microsoft.VisualStudio.Workload.ManagedDesktop --add Component.GitHub.VisualStudio --  
includeRecommended --lang en-US de-DE ja-JP
```

- To download two workloads and all of their recommended components:

```
vs_enterprise.exe --layout C:\VSLayout --add Microsoft.VisualStudio.Workload.Azure --add  
Microsoft.VisualStudio.Workload.ManagedDesktop --add Component.GitHub.VisualStudio --  
includeRecommended
```

- To download two workloads and all of their recommended and optional components, run:

```
vs_enterprise.exe --layout C:\VSLayout --add Microsoft.VisualStudio.Workload.Azure --add  
Microsoft.VisualStudio.Workload.ManagedDesktop --add Component.GitHub.VisualStudio --includeOptional
```

New in version 15.3

Save your layout options

When you run a layout command, the options that you specify are saved (such as the workloads and languages). Subsequent layout commands will include all of the previous options. Here is an example of a layout with one workload for English only:

```
vs_enterprise.exe --layout c:\VSLayout --add Microsoft.VisualStudio.Workload.ManagedDesktop --lang en-US
```

When you want to update that layout to a newer version, you don't have to specify any additional command-line parameters. The previous settings are saved and used by any subsequent layout commands in this layout folder. The following command will update the existing partial layout.

```
vs_enterprise.exe --layout c:\VSLayout
```

When you want to add an additional workload, here's an example of how to do so. In this case, we'll add the Azure workload and a localized language. Now, both Managed Desktop and Azure are included in this layout. The language resources for English and German are included for all these workloads. The layout is updated to the latest available version.

```
vs_enterprise.exe --layout c:\VSLayout --add Microsoft.VisualStudio.Workload.Azure --lang de-DE
```

If you want to update an existing layout to a full layout, use the --all option, as shown in the following example.

```
vs_enterprise.exe --layout c:\VSLayout --all
```

Deploy from a network installation

Administrators can deploy Visual Studio onto client workstations as part of an installation script. Or, users who have administrator rights can run setup directly from the share to install Visual Studio on their machine.

- Users can install by running the following command:

```
\server\products\VS\vs_enterprise.exe
```

- Administrators can install in an unattended mode by running the following command:

```
\server\products\VS\vs_enterprise.exe --quiet --wait --norestart
```

IMPORTANT

To prevent an error, make sure that your full layout path is less than 80 characters.

TIP

When executed as part of a batch file, the `--wait` option ensures that the `vs_enterprise.exe` process waits until the installation is complete before it returns an exit code.

This is useful if an enterprise administrator wants to perform further actions on a completed installation (for example, to [apply a product key to a successful installation](#)) but must wait for the installation to finish to handle the return code from that installation.

If you do not use `--wait`, the `vs_enterprise.exe` process exits before the installation is complete and returns an inaccurate exit code that doesn't represent the state of the install operation.

IMPORTANT

For offline installations, if you get an error message that says "A product matching the following parameters cannot be found", make sure that you are using the `--noWeb` switch with version 16.3.5 or later.

When you install from a layout, the content that is installed is acquired from the layout. However, if you select a component that isn't in the layout, it will be acquired from the internet. If you want to prevent Visual Studio setup from downloading any content that is missing in your layout, use the `--noWeb` option. If `--noWeb` is used and the layout is missing any content that is selected to be installed, setup fails.

IMPORTANT

The `--noWeb` option does not stop Visual Studio setup from checking for updates. For more information, see the [Control updates to network-based Visual Studio deployments](#) page.

Error codes

If you used the `--wait` parameter, then depending on the result of the operation, the `%ERRORLEVEL%` environment variable is set to one of the following values:

VALUE	RESULT
0	Operation completed successfully
1602	Operation was canceled
1641	Operation completed successfully, and reboot was initiated
3010	Operation completed successfully, but install requires reboot before it can be used
5003	Bootstrapper failed to download installer
5004	Operation was canceled
5005	Bootstrapper command-line parse error
5007	Operation was blocked - the computer does not meet the requirements

VALUE	RESULT
-1073741510	Microsoft Visual Studio Installer was terminated (by the user or external process)
Other (for example: -1, 1, 1603)	Failure condition occurred - check the logs for more information

Update a network install layout

As product updates become available, you might want to [update the network install layout](#) to incorporate updated packages.

How to create a layout for a previous Visual Studio release

NOTE

The Visual Studio bootstrappers that are available on visualstudio.microsoft.com download and install the latest Visual Studio release that's available whenever they are run.

So, if you download a Visual Studio *bootstrapper* today and run it six months from now, it installs the Visual Studio release that is current at the time you run the bootstrapper.

But, if you create a *layout* and then install from it, the layout installs the specific version of Visual Studio that exists in the layout. Even though a newer version might exist online, you get the version of Visual Studio that is in the layout.

NOTE

The Visual Studio bootstrappers that are available on visualstudio.microsoft.com download and install the latest Visual Studio release that's available whenever they are run.

So, if you download a Visual Studio *bootstrapper* today and run it six months from now, it installs the Visual Studio release that is current at the time you run the bootstrapper.

But, if you create a *layout* and then install from it, the layout installs the specific version of Visual Studio that exists in the layout. Even though a newer version might exist online, you get the version of Visual Studio that is in the layout.

If you need to create a layout for an older version of Visual Studio, go to <https://my.visualstudio.com> to download "fixed" versions of the Visual Studio bootstrappers.

How to get support for your offline installer

If you experience a problem with your offline installation, we want to know about it. The best way to tell us is by using the [Report a Problem](#) tool. When you use this tool, you can send us the telemetry and logs we need to help us diagnose and fix the problem.

We also offer a [live chat](#) (English only) support option for installation-related issues.

We have other support options available, too. For a list, see our [Feedback](#) page.

See also

- [Visual Studio administrator guide](#)
- [Update a network-based installation of Visual Studio](#)
- [Troubleshoot network-related errors when you install or use Visual Studio](#)

- Control updates to network-based Visual Studio deployments
- Visual Studio product lifecycle and servicing
- Update Visual Studio while on a servicing baseline
- Use command-line parameters to install Visual Studio
- Visual Studio workload and component IDs

Install and use Visual Studio and Azure Services behind a firewall or proxy server

10/31/2019 • 10 minutes to read • [Edit Online](#)

If you or your organization uses security measures such as a firewall or a proxy server, then there are domain URLs that you might want to add to an "allow list" and ports and protocols that you might want to open so that you have the best experience when you install and use Visual Studio and Azure Services.

- **Install Visual Studio:** These tables include the domain URLs to add to an allow list so that you have access to all the components and workloads that you want.
- **Use Visual Studio and Azure Services:** This table includes the domain URLs to add to an allow list and the ports and protocols to open so that you have access to all the features and services that you want.

NOTE

This article was written for Visual Studio on Windows, but certain information is also applicable to [installing Visual Studio for Mac](#) behind a firewall or proxy server.

Install Visual Studio

URLs to add to an allow list

Because the Visual Studio Installer downloads files from various domains and their download servers, here are the domain URLs that you might want to add to an allow list as trusted in the UI or in your deployment scripts.

Microsoft domains

DOMAIN	PURPOSE
go.microsoft.com	Setup URL resolution
aka.ms	Setup URL resolution
download.visualstudio.microsoft.com	Setup packages download location
download.microsoft.com	Setup packages download location
download.visualstudio.com	Setup packages download location
dl.xamarin.com	Setup packages download location
xamarin-downloads.azureedge.net	Android SDK packages download list location
marketplace.visualstudio.com	Visual Studio Extensions download location
visualstudio.microsoft.com	Documentation location
docs.microsoft.com	Documentation location

DOMAIN	PURPOSE
msdn.microsoft.com	Documentation location
www.microsoft.com	Documentation location
*.windows.net	Sign-in location
*.microsoftonline.com	Sign-in location
*.live.com	Sign-in location

Non-Microsoft domains

DOMAIN	INSTALLS THESE WORKLOADS
archive.apache.org	Mobile development with JavaScript (Cordova)
cocos2d-x.org	Game development with C++ (Cocos)
download.epicgames.com	Game development with C++ (Unreal Engine)
download.oracle.com	Mobile development with JavaScript (Java SDK) Mobile Development with .NET (Java SDK)
download.unity3d.com	Game development with Unity (Unity)
netstorage.unity3d.com	Game development with Unity (Unity)
dl.google.com	Mobile development with JavaScript (Android SDK and NDK, Emulator) Mobile Development with .NET (Android SDK and NDK, Emulator)
www.incredibuild.com	Game development with C++ (Incredibuild)
incredibuildvs2017.azureedge.net	Game development with C++ (Incredibuild)
www.python.org	Python development (Python) Data science and analytical applications (Python)

Use Visual Studio and Azure Services

URLs to add to an allow list and ports and protocols to open

To make sure that you have access to everything you need when you use Visual Studio or Azure Services behind a firewall or proxy server, here are the URLs you should add to an allow list and the ports and protocols that you might want to open.

Service or Scenario	DNS Endpoint	Protocol	Port	Description
URL resolution	go.microsoft.com aka.ms			Used to shorten URLs, which then resolve into longer URLs
Start Page	vsstartpage.blob.core.windows.net		443	Used to display Developer News shown on the start page (Visual Studio 2017 only)
Targeted Notification Service	targetednotifications.azurewebsites.net www.research.net		80 443	Used to filter a global list of notifications to a list that is applicable only to specific types of machines/usage scenarios
Extension update check	marketplace.visualstudio.com *.windows.net *.microsoftonline.com *.live.com		443	Used to provide notifications when an installed extension has an update available Used as a sign-in location
AI Project Integration	az861674.vo.msecnd.net		443	Used to configure new projects to send usage data to your registered Application Insights account
Code Lens	codelensprodscus1su0.app.codelens.visualstudio.com		443	Used to provide information in the editor about when a file was last updated, the timeline of changes, the work items that changes are associated with, the authors, and more
Experimental feature enabling	visualstudio-devdiv-c2s.msedge.net		80	Used to activate experimental new features or feature changes

Service or Scenario	DNS Endpoint	Protocol	Port	Description
Identity "badge" (user name and avatar) and Roaming settings	app.vssps.visualstudio.com app.vsspsext.visualstudio.com app.vssps.visualstudio.com ns-sb2-prod-ch1-002.cloudapp.net az700632.vo.msecnd.net		443	Used to display the user's name and avatar in the IDE Used to make sure that setting changes roam from one machine to another
Remote Settings	az700632.vo.msecnd.net		443	Used to turn off extensions that are known to cause problems in Visual Studio
Windows Tools	developer.microsoft.com dev.windows.com appdev.microsoft.com	https	443	Used for Windows app store scenarios
JSON Schema Discovery JSON Schema Definition JSON Schema Support for Azure Resources	json.schemastore.org schemastoreorg.azurewebsites.net json-schema.org schema.management.azure.com	http https http https	80 443 443 443	Used to discover and download JSON schemas that the user might use when editing JSON documents Used to obtain the meta-validation schema for JSON Used to obtain the current schema for Azure Resource Manager deployment templates
NPM package discovery	Skimdb.npmjs.com Registry.npmjs.org Api.npms.io	https http/s https	443 80/443 443	Required for searching for NPM packages, and used for client-side script package installation in web projects
Bower package icons Bower package search	Bower.io bowercache.azurewebsites.net go.microsoft.com Registry.bower.io	http https http https	80 443 80 443	Provides the default bower package icon Provides the ability to search for Bower packages

Service or Scenario	DNS Endpoint	Protocol	Port	Description
NuGet NuGet package discovery	Api.nuget.org www.nuget.org Nuget.org crl3.digicert.com crl4.digicert.com ocsp.digicert.com cacerts.digicert.com	https http/s	443 80/443	Used to verify signed NuGet packages. Required for searching for NuGet packages and versions
GitHub repository information	api.github.com	https	443	Required for getting additional information about bower packages
Web Linters	Eslint.org www.Bing.com www.coffeelint.org	http	80	
Cookiecutter Explorer template discovery Cookiecutter Explorer project creation	api.github.com raw.githubusercontent.com go.microsoft.com pypi.org pypi.python.org	https	443	Used to discover online templates from our recommended feed and from GitHub repositories Used to create a project from a cookiecutter template that requires a one-time on-demand installation of a cookiecutter Python package from the Python package index (PyPI)
Python package discovery Python package management New Python project templates	pypi.org pypi.python.org bootstrap.pypa.io go.microsoft.com	https	443	Provides the ability to search for pip packages Used to install pip automatically if it is missing Used to resolve the following new Python project templates to cookiecutter template URLs: - Classifier Project - Clustering Project - Regression Project - PyGame using PyKinect - Pivot Project

Service or Scenario	DNS Endpoint	Protocol	Port	Description
Office web add-in Manifest Verification Service	verificationservice.osi.office.net	https	443	Used to validate manifests for Office web add-ins
SharePoint and Office Add-ins	sharepoint.com	https	443	Used to publish and test SharePoint and Office Add-ins to SharePoint Online
Workflow Manager Test Service Host		http	12292	A firewall rule that is created automatically for testing SharePoint add-ins with workflows
Automatically collected reliability statistics and other Customer Experience Improvement Programs (CEIP) for Azure SDK and for SQL Tools	vortex.data.microsoft.com dc.services.visualstudio.com	https	443	Used to send reliability statistics (crash/hang data) from the user to Microsoft. Actual crash/hang dumps will still be uploaded if Windows Error Reporting is enabled; only statistical information will be suppressed; Used to reveal anonymous usage patterns for the Azure Tools SDK extension to Visual Studio, and for usage patterns for the SQL tooling to Visual Studio
Visual Studio Customer Experience Improvement Program (CEIP) PerfWatson.exe	vortex.data.microsoft.com dc.services.visualstudio.com visualstudio-devdiv-c2s.msedge.net az667904.vo.msecnd.net scus-breeziest-in.cloudapp.net	https	443	Used to collect anonymous usage patterns and error logs Used to track UI freeze issues
Creation and Management of Azure resources	management.azure.com management.core.windows.net	https	443	Used for creating Azure Websites or other resources to support the publishing of web applications, Azure Functions, or WebJobs

Service or Scenario	DNS Endpoint	Protocol	Port	Description
Updated web publish tooling checks and extension recommendations	marketplace.visualstudio.com	https	443	Used for checking for the availability of updated publish tooling. If disabled, a potential recommended extension for web publishing may not be shown
Updated Azure Resource Creation Endpoint Information	*.blob.core.windows.net	https	443	Used to update the endpoints used for the creation of Azure Resources for certain Azure Services. If disabled, the last downloaded or built in endpoint locations are used instead
Remote debugging and Remote profiling of Azure Websites	*.cloudapp.net *.azurewebsites.net		4022	Used for attaching the remote debugger to Azure Websites. If disabled, attaching the remote debugger to Azure Websites will not work
Active Directory Graph	graph.windows.net	https	443	Used to provision new Azure Active Directory applications. Also used by the Office 365 MSGraph-connected service provider
Azure Functions CLI Update Check	functionscdn.azureedge.net	https	443	Used for checking for updated versions of the Azure Functions CLI. If disabled, a cached copy (or the copy carried by the Azure Functions component) of the CLI will be used instead
Cordova	npmjs.org gradle.org	http/s	80/443	HTTP is used for Gradle downloads during build; HTTPS is used to include Cordova plug-ins in projects

Service or Scenario	DNS Endpoint	Protocol	Port	Description
Cloud explorer	1. <clusterendpoint> Service Fabric 2. <management endpoint> General Cloud Exp 3. <graph endpoint> General Cloud Exp 4. <storage account endpoint> Storage Nodes 5. <Azure portal URLs> General Cloud Exp 6. <key vault endpoints> Azure Resource Manager VM Nodes 7. <PublicIPAddressOfCluster> Service Fabric Remote debugging and ETW Traces	1. https 2. https 3. https 4. https 5. https 6. https 7: tcp	1. 19080 2. 443 3. 443 4. 443 5. 443 6. 443 7. dynamic	<p>1. Example: test12.eastus.cloudapp.com</p> <p>2. Retrieves subscriptions and retrieves/manages Azure resources</p> <p>3. Retrieves Azure Stack subscriptions</p> <p>4. Manages Storage resources (example: mystorageaccount.blob.core.windows.net)</p> <p>5. "Open in Portal" context menu option (opens a resource in the Azure portal)</p> <p>6. Creates and uses key vaults for VM debugging (Example: myvault.vault.azure.net)</p> <p>7. Dynamically allocates block of ports based on number of nodes in the cluster and the available ports.</p> <p>A port block will try to get three times the number of nodes with minimum of 10 ports.</p> <p>For Streaming traces, an attempt is made to get the port block from 810. If any of that port block is already used, then an attempt is made to get the next block, and so on. (If the load balancer is empty, then ports from 810 are most likely used)</p> <p>Similarly for debugging, four sets of the ports blocks are reserved:</p> <ul style="list-style-type: none"> - connectorPort: 30398, - forwarderPort: 31398, - forwarderPortx86: 31399, - fileUploadPort: 32398

Service or Scenario	DNS Endpoint	Protocol	Port	Description
Cloud Services	1. RDP 2. core.windows.net 3. management.azure.com management.core.windows.net 4. *.blob.core.windows.net *.queue.core.windows.net *.table.core.windows.net 5. portal.azure.com 6. <user's cloud service>.cloudapp.net <user's VM>. <region>.azure.com	1. rdp 2. https 3. https 4. https 5. https 6. tcp	1. 3389 2. 443 3. 443 4. 443 5. 443 6. a) 30398 6. b) 30400 6. c) 31398 6. d) 31400 6. e) 32398 6. f) 32400	1. Remote Desktop to Cloud Services VM 2. Storage account component of the private diagnostics configuration 3. Azure portal 4. Server Explorer - Azure Storage * is customer named storage account 5. Links to open the portal / Download the subscription certificate / Publish settings file 6. a) Connector local port for remote debug for cloud service and VM 6. b) Connector public port for remote debug for cloud service and VM 6. c) Forwarder local port for remote debug for cloud service and VM 6. d) Forwarder public port for remote debug for cloud service and VM 6. e) File uploader local port for remote debug for cloud service and VM 6. f) File uploader public port for remote debug for cloud service and VM

Service or Scenario	DNS Endpoint	Protocol	Port	Description
Service Fabric	1. ocs.Microsoft.com aka.ms go.microsoft.com 2. vssftools.blob.core.windows.net Vault.azure.com Portal.azure.com 3. * vault.azure.net 4. app.vsaex.visualstudio.com * .vsspsext.visualstudio.com clouds.vsrm.visualstudio.io.com clouds.visualstudio.com app.vssps.visualstudio.com * .visualstudio.com	https	443	1. Documentation 2. Create Cluster feature 3. The * is the Azure key vault name (Example:- test11220180112110 108.vault.azure.net) 4. The * is dynamic (Example: vsspsextprodch1su1.vsspsext.visualstudio.com)
Snapshot Debugger	1. go.microsoft.com 2. management.azure.com 3. *azurewebsites.net 4. *scm.azurewebsites.net 5. api.nuget.org/v3/index.json 6. msvsmon (.exe)	1. https 2. https 3. http 4. https 5. https 6. Concord	1. 443 2. 443 3. 80 4. 443 5. 443 6. 4022 (Visual Studio version dependent)	1. Query .json file for app service SKU size 2. Various Azure RM calls 3. Site warmup call via 4. Customer's targeted App Service Kudu endpoint 5. Query Site Extension version published in nuget.org 6. Remote debugging channel
Azure Stream Analytics HDInsight	Management.azure.com	https	443	Used to view, submit, run, and manage ASA jobs Used to browse HDI clusters, and to submit, diagnose, and debug HDI jobs
Azure Data Lake	*.azuredatalakestore.net *.azuredatalakeanalyticstcs.net	https	443	Used to compile, submit, view, diagnose, and debug jobs; used to browse ADLS files; used to upload and download files

Service or scenario	DNS endpoint	Protocol	Port	Description
Packaging Service	[account].visualstudio.com [account].*.visualstudio.com .blob.core.windows.net registry.npmjs.org nodejs.org dist.nuget.org nuget.org	https	443	The *.npmjs.org, *.nuget.org, and *.nodejs.org are only required for certain build task scenarios (for example: NuGet Tool Installer, Node Tool Installer) or if you intend to use public upstreams with your Feeds. The other three domains are required for core functionality of the Packaging service.
Azure DevOps Services	*.vsassets.io static2.sharepointonline.com dev.azure.com			Used to connect with Azure DevOps Services

Troubleshoot network-related errors

Sometimes, you might run in to network- or proxy-related errors when you install or use Visual Studio behind a firewall or a proxy server. For more information about solutions for such error messages, see the [Troubleshooting network-related errors when you install or use Visual Studio](#) page.

Get support

We offer a [live chat](#) (English only) support option for installation-related issues.

Here are a few more support options:

- Report product issues to us via the [Report a Problem](#) tool that appears both in the Visual Studio Installer and in the Visual Studio IDE.
- Suggest a feature, track product issues, and find answers in the [Visual Studio Developer Community](#).
- Use your [GitHub](#) account to talk to us and other Visual Studio developers in the [Visual Studio conversation in the Gitter community](#).

See also

- [Connectivity requirements for Live Share](#)
- [Create a network installation of Visual Studio](#)
- [Troubleshoot network-related errors in Visual Studio](#)
- [Visual Studio administrator guide](#)
- [Install behind a firewall or proxy server \(Visual Studio for Mac\)](#)

Troubleshoot network-related errors when you install or use Visual Studio

11/7/2019 • 4 minutes to read • [Edit Online](#)

We've got solutions for the most typical network- or proxy-related errors that you might encounter when you install or use Visual Studio behind a firewall or a proxy server.

Error: "Proxy authorization required"

This error generally occurs when users are connected to the internet through a proxy server, and the proxy server blocks the calls that Visual Studio makes to some network resources.

To fix this proxy error

- Restart Visual Studio. A proxy authentication dialog box should appear. Enter your credentials when prompted in the dialog.
- If restarting Visual Studio does not solve the problem, it might be that your proxy server does not prompt for credentials for `http://go.microsoft.com` addresses but does so for `*.visualstudio.microsoft.com` addresses. For these servers, consider adding the following URLs to an allow list to unblock all sign-in scenarios in Visual Studio:
 - `*.windows.net`
 - `*.microsoftonline.com`
 - `*.visualstudio.microsoft.com`
 - `*.microsoft.com`
 - `*.live.com`

- You can otherwise remove the `http://go.microsoft.com` address from the allow list so that the proxy authentication dialog shows up for both the `http://go.microsoft.com` address and the server endpoints when Visual Studio is restarted.

-OR-

- If you want to use your default credentials with your proxy, you can perform the following actions:
 1. Find **devenv.exe.config** (the devenv.exe configuration file) in: **%ProgramFiles%\Microsoft Visual Studio\2017\Enterprise\Common7\IDE** or **%ProgramFiles(x86)%\Microsoft Visual Studio\2017\Enterprise\Common7\IDE**.
 2. In the configuration file, find the `<system.net>` block, and then add this code:

```
<defaultProxy enabled="true" useDefaultCredentials="true">
  <proxy bypassOnLocal="True" proxyAddress="http://<yourproxy:port#>"/>
</defaultProxy>
```

You must insert the correct proxy address for your network in `proxyAddress="<http://<yourproxy:port#>"`.

NOTE

For more information, see the [<defaultProxy> Element \(Network Settings\)](#) and [<proxy> Element \(Network Settings\)](#) pages.

1. Find **devenv.exe.config** (the devenv.exe configuration file) in: **%ProgramFiles%\Microsoft Visual Studio\2019\Enterprise\Common7\IDE** or **%ProgramFiles(x86)%\Microsoft Visual Studio\2019\Enterprise\Common7\IDE**.
2. In the configuration file, find the **<system.net>** block, and then add this code:

```
<defaultProxy enabled="true" useDefaultCredentials="true">
    <proxy bypassOnLocal="True" proxyaddress="http://<yourproxy:port#>"/>
</defaultProxy>
```

You must insert the correct proxy address for your network in **proxyaddress="<http://<yourproxy:port#>"**.

NOTE

For more information, see the [<defaultProxy> Element \(Network Settings\)](#) and [<proxy> Element \(Network Settings\)](#) pages.

Error: “The underlying connection was closed”

If you are using Visual Studio in a private network that has a firewall, Visual Studio might not be able to connect to some network resources. These resources can include Azure DevOps Services for sign-in and licensing, NuGet, and Azure services. If Visual Studio fails to connect to one of these resources, you might see the following error message:

The underlying connection was closed: An unexpected error occurred on send

Visual Studio uses Transport Layer Security (TLS) 1.2 protocol to connect to network resources. Security appliances on some private networks block certain server connections when Visual Studio uses TLS 1.2.

To fix this connection error

Enable connections for the following URLs:

- <https://management.core.windows.net>
- <https://app.vssps.visualstudio.com>
- <https://login.microsoftonline.com>
- <https://login.live.com>
- <https://go.microsoft.com>
- <https://graph.windows.net>
- <https://app.vsspsext.visualstudio.com>
- *.azurewebsites.net (for Azure connections)
- *.visualstudio.microsoft.com
- cdn.vsassets.io (hosts content delivery network, or CDN, content)
- *.gallerycdn.vsassets.io (hosts Azure DevOps Services extensions)

- static2.sharepointonline.com (hosts resources that Visual Studio uses in the Office UI Fabric kit, such as fonts)
- *.nuget.org (for NuGet connections)

NOTE

Privately owned NuGet server URLs may not be included in this list. You can check for the NuGet servers that you are using in %APPData%\Nuget\NuGet.Config.

Error: "Failed to parse ID from parent process"

You might encounter this error message when you use a Visual Studio bootstrapper and a response.json file on a network drive. The error's source is the User Account Control (UAC) in Windows.

Here's why this error can happen: A mapped network drive or [UNC](#) share is linked to a user's access token. When UAC is enabled, two user [access tokens](#) are created: One *with* administrator access, and one *without* administrator access. When a network drive or share is created, the user's current access token is linked to it. Because the bootstrapper must be run as administrator, it won't be able to access the network drive or share if either the drive or the share isn't linked to a user access token that has administrator access.

To fix this error

You can use the `net use` command or you can change the UAC Group Policy setting. For more information about these workarounds and how to implement them, see the following Microsoft support articles:

- Mapped drives are not available from an elevated prompt when UAC is configured to "Prompt for credentials" in Windows
- Programs may be unable to access some network locations after you turn on User Account Control in Windows operating systems

Get support

Sometimes, things can go wrong. If your Visual Studio installation fails, see [Troubleshoot Visual Studio installation and upgrade issues](#) for step-by-step guidance.

We also offer a [live chat](#) (English only) support option for installation-related issues.

Here are a few more support options:

- Report product issues to us via the [Report a Problem](#) tool that appears both in the Visual Studio Installer and in the Visual Studio IDE.
- Suggest a feature, track product issues, and find answers in the [Visual Studio Developer Community](#).
- Use your [GitHub](#) account to talk to us and other Visual Studio developers in the [Visual Studio conversation in the Gitter community](#).

See also

- [Install and use Visual Studio behind a firewall or proxy server](#)
- [Visual Studio administrator guide](#)
- [Install Visual Studio](#)

How to define settings in a response file

10/31/2019 • 3 minutes to read • [Edit Online](#)

Administrators who deploy Visual Studio can specify a response file by using the `--in` parameter, as in the following example:

```
vs_enterprise.exe --in customInstall.json
```

Response files are [JSON](#) files whose contents mirror the command-line arguments. In general, if a command-line parameter takes no arguments (for example, `--quiet`, `--passive`, etc.), the value in the response file should be true/false. If it takes an argument (for example, `--installPath <dir>`), the value in the response file should be a string. If it takes an argument and can appear on the command line more than once (for example, `--add <id>`), it should be an array of strings.

Parameters that are specified on the command-line override settings from the response file, except when parameters take multiple inputs (for example, `--add`). When you have multiple inputs, the inputs supplied on the command line are merged with settings from the response file.

Setting a default configuration for Visual Studio

If you created a network layout cache with the `--layout`, an initial `response.json` file is created in the layout. If you create a partial layout, this response file includes the workloads and languages that were included in the layout. Running setup from this layout automatically uses this `response.json` file, which selects the workloads and components included in the layout. Users can still select or unselect any workloads in the setup UI before installing Visual Studio.

Administrators who create a layout can modify the `response.json` file in the layout to control the default settings that their users see when they install Visual Studio from the layout. For example, if an administrator wants specific workloads and components installed by default, they can configure the `response.json` file to add them.

When Visual Studio setup is run from a layout folder, it *automatically* uses the response file in the layout folder. You don't have to use the `--in` option.

You can update the `response.json` file that is created in an offline layout folder to define the default setting for users who install from this layout.

WARNING

It's critical that you leave the existing properties that were defined when the layout was created.

The base `response.json` file in a layout should look similar to the following example, except that it would include the value for the product and channel that you want to install:

```
{
  "installChannelUri": ".\\ChannelManifest.json",
  "channelUri": "https://aka.ms/vs/15/release/channel",
  "installCatalogUri": ".\\Catalog.json",
  "channelId": "VisualStudio.15.Release",
  "productId": "Microsoft.VisualStudio.Product.Enterprise"
}
```

```
{
  "installChannelUri": ".\\ChannelManifest.json",
  "channelUri": "https://aka.ms/vs/16/release/channel",
  "installCatalogUri": ".\\Catalog.json",
  "channelId": "VisualStudio.16.Release",
  "productId": "Microsoft.VisualStudio.Product.Enterprise"
}
```

When you create or update a layout, a response.template.json file is also created. This file contains all of the workload, component, and language IDs that can be used. This file is provided as a template for what all could be included in a custom install. Administrators can use this file as a starting point for a custom response file. Just remove the IDs for the things you do not want to install and save it in your own response file. Do not customize the response.template.json file or your changes will be lost whenever the layout is updated.

Example layout response file content

The following example installs Visual Studio Enterprise with six common workloads and components, and with both English and French UI languages. You can use this example as a template; just change the workloads and components to those that you want to install:

```
{
  "installChannelUri": ".\\ChannelManifest.json",
  "channelUri": "https://aka.ms/vs/15/release/channel",
  "installCatalogUri": ".\\Catalog.json",
  "channelId": "VisualStudio.15.Release",
  "productId": "Microsoft.VisualStudio.Product.Enterprise",

  "installPath": "C:\\VS2017",
  "quiet": false,
  "passive": false,
  "includeRecommended": true,
  "norestart": false,

  "addProductLang": [
    "en-US",
    "fr-FR"
  ],

  "add": [
    "Microsoft.VisualStudio.Workload.ManagedDesktop",
    "Microsoft.VisualStudio.Workload.Data",
    "Microsoft.VisualStudio.Workload.NativeDesktop",
    "Microsoft.VisualStudio.Workload.NetWeb",
    "Microsoft.VisualStudio.Workload.Office",
    "Microsoft.VisualStudio.Workload.Universal",
    "Component.GitHub.VisualStudio"
  ]
}
```

```
{  
  "installChannelUri": ".\\ChannelManifest.json",  
  "channelUri": "https://aka.ms/vs/16/release/channel",  
  "installCatalogUri": ".\\Catalog.json",  
  "channelId": "VisualStudio.16.Release",  
  "productId": "Microsoft.VisualStudio.Product.Enterprise",  
  
  "installPath": "C:\\VS2019",  
  "quiet": false,  
  "passive": false,  
  "includeRecommended": true,  
  "norestart": false,  
  
  "addProductLang": [  
    "en-US",  
    "fr-FR"  
  ],  
  
  "add": [  
    "Microsoft.VisualStudio.Workload.ManagedDesktop",  
    "Microsoft.VisualStudio.Workload.Data",  
    "Microsoft.VisualStudio.Workload.NativeDesktop",  
    "Microsoft.VisualStudio.Workload.NetWeb",  
    "Microsoft.VisualStudio.Workload.Office",  
    "Microsoft.VisualStudio.Workload.Universal",  
    "Component.GitHub.VisualStudio"  
  ]  
}
```

Get support

Sometimes, things can go wrong. If your Visual Studio installation fails, see [Troubleshoot Visual Studio installation and upgrade issues](#) for step-by-step guidance.

We also offer a [live chat](#) (English only) support option for installation-related issues.

Here are a few more support options:

- Report product issues to us via the [Report a Problem](#) tool that appears both in the Visual Studio Installer and in the Visual Studio IDE.
- Suggest a feature, track product issues, and find answers in the [Visual Studio Developer Community](#).
- Use your [GitHub](#) account to talk to us and other Visual Studio developers in the [Visual Studio conversation in the Gitter community](#).

See also

- [Visual Studio workload and component IDs](#)
- [Troubleshoot network-related errors when you install or use Visual Studio](#)

Automatically apply product keys when deploying Visual Studio

10/2/2019 • 2 minutes to read • [Edit Online](#)

You can apply your product key programmatically as part of a script that is used to automate the deployment of Visual Studio. You can set a product key on a device programmatically either during an installation of Visual Studio or after an installation completes.

Apply the license after installation

You can activate an installed version of Visual Studio with a product key by using the `StorePID.exe` utility on the target machines, in silent mode. `StorePID.exe` is a utility program that installs with Visual Studio 2017 at the following default location:

```
C:\Program Files (x86)\Microsoft Visual Studio\2017\Enterprise\Common7\IDE
```

You can activate an installed version of Visual Studio with a product key by using the `StorePID.exe` utility on the target machines, in silent mode. `StorePID.exe` is a utility program that installs with Visual Studio 2019 at the following default location:

```
C:\Program Files (x86)\Microsoft Visual Studio\2019\Enterprise\Common7\IDE
```

Run `StorePID.exe` with elevated privileges, either by using a System Center agent or an elevated command prompt. Follow it with the product key and the Microsoft Product Code (MPC).

IMPORTANT

Make sure to include the dashes in the product key.

```
StorePID.exe [product key including the dashes] [MPC]
```

The following example shows a command line for applying the license for Visual Studio 2017 Enterprise, which has an MPC of 08860, a product key of `AAAAAA-BBBBBB-CCCCCC-DDDDDD-EEEEEEE`, and assumes a default installation location:

```
"C:\Program Files (x86)\Microsoft Visual Studio\2017\Enterprise\Common7\IDE\StorePID.exe" AAAAAA-BBBBBB-CCCCCC-DDDDDD-EEEEEEE 08860
```

The following example shows a command line for applying the license for Visual Studio 2019 Enterprise, which has an MPC of 09260, a product key of `AAAAAA-BBBBBB-CCCCCC-DDDDDD-EEEEEEE`, and assumes a default installation location:

```
"C:\Program Files (x86)\Microsoft Visual Studio\2019\Enterprise\Common7\IDE\StorePID.exe" AAAAAA-BBBBBB-CCCCCC-DDDDDD-EEEEEEE 09260
```

The following table lists the MPC codes for each edition of Visual Studio:

VISUAL STUDIO EDITION	MPC
Visual Studio Enterprise 2017	08860
Visual Studio Professional 2017	08862
Visual Studio Test Professional 2017	08866
VISUAL STUDIO EDITION	MPC
Visual Studio Enterprise 2019	09260
Visual Studio Professional 2019	09262

If `StorePID.exe` successfully applies the product key, it returns an `%ERRORLEVEL%` of 0. If it encounters errors, it returns one of the following codes, depending on the error condition:

ERROR	CODE
<code>PID_ACTION_SUCCESS</code>	0
<code>PID_ACTION_NOTINSTALLED</code>	1
<code>PID_ACTION_INVALID</code>	2
<code>PID_ACTION_EXPIRED</code>	3
<code>PID_ACTION_INUSE</code>	4
<code>PID_ACTION_FAILURE</code>	5
<code>PID_ACTION_NOUPGRADE</code>	6

NOTE

When you run a virtual instance of Visual Studio, make sure that you also virtualize the local AppData folder and the registry. To troubleshoot virtual instances, run

```
C:\Program Files (x86)\Microsoft Visual Studio\<version>\Common7\IDE\DDConfigCA.exe
```

Get support

Sometimes, things can go wrong. If your Visual Studio installation fails, see [Troubleshoot Visual Studio installation and upgrade issues](#) for step-by-step guidance.

We also offer a [live chat](#) (English only) support option for installation-related issues.

Here are a few more support options:

- Report product issues to us via the [Report a Problem](#) tool that appears both in the Visual Studio Installer and in the Visual Studio IDE.
- Suggest a feature, track product issues, and find answers in the [Visual Studio Developer Community](#).
- Use your [GitHub](#) account to talk to us and other Visual Studio developers in the [Visual Studio conversation](#) in

[the Gitter community.](#)

See also

- [Install Visual Studio](#)
- [Create an offline installation of Visual Studio](#)

Set defaults for enterprise deployments of Visual Studio

8/16/2019 • 2 minutes to read • [Edit Online](#)

You can set registry policies that affect the deployment of Visual Studio. These policies are global for the new installer and affect:

- Where some packages shared with other versions or instances are installed
- Where packages are cached
- Whether all packages are cached

You can set some of these policies using [command-line options](#), set registry values on your machine, or even distribute them using Group Policy across an organization.

Registry keys

There are several locations where you can set enterprise defaults, to enable their control either through Group Policy or directly in the registry. Visual Studio looks sequentially to see if any enterprise policies have been set; as soon as a policy value is discovered in the order below, the remaining keys are ignored.

1. `HKEY_LOCAL_MACHINE\SOFTWARE\Policies\Microsoft\VisualStudio\Setup`
2. `HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\VisualStudio\Setup`
3. `HKEY_LOCAL_MACHINE\SOFTWARE\WOW6432Node\Microsoft\VisualStudio\Setup` (on 64-bit operating systems)

IMPORTANT

If you do not set the `HKEY_LOCAL_MACHINE\SOFTWARE\Policies\Microsoft\VisualStudio\Setup` key and instead set one of the other keys, you should set both other keys on 64-bit operating systems. This issue is addressed in a future product update.

Some registry values are set automatically the first time they are used if not set already. This practice ensures that subsequent installs use the same values. These values are stored in the second registry key,

`HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\VisualStudio\Setup`.

You can set the following registry values:

NAME	TYPE	DEFAULT	DESCRIPTION
<code>CachePath</code>	<code>REG_SZ</code> or <code>REG_EXPAND_SZ</code>	<code>%ProgramData%\Microsoft\VisualStudio\Packages</code>	The directory where package manifests and, optionally, payloads are stored. For more information, see the Disable or move the package cache page.

NAME	TYPE	DEFAULT	DESCRIPTION
KeepDownloadedPayloads	REG_DWORD	1	Keep package payloads even after they are installed. You can change the value anytime. Disabling the policy removes any cached package payloads for the instance you repair or modify. For more information, see the Disable or move the package cache page.
SharedInstallationPath	REG_SZ or REG_EXPAND_SZ	%ProgramFiles(x86)%\Microsoft Visual Studio\Shared	The directory where some packages shared across versions of instances of Visual Studio are installed. You can change the value any time, but it will only affect future installs. Any products already installed to the old location must not be moved or they might not function correctly.
BackgroundDownloadDisabled	REG_DWORD	1	Prevent setup from downloading updates automatically for all installed Visual Studio products. You can change the value anytime.

IMPORTANT

If you change the `CachePath` registry policy after any installations, you must move the existing package cache to the new location and make sure it's secured so that `SYSTEM` and `Administrators` have Full Control and that `Everyone` has Read access. Failure to move the existing cache or securing it might cause problems with future installs.

Get support

Sometimes, things can go wrong. If your Visual Studio installation fails, see [Troubleshoot Visual Studio installation and upgrade issues](#) for step-by-step guidance.

We also offer a [live chat](#) (English only) support option for installation-related issues.

Here are a few more support options:

- Report product issues to us via the [Report a Problem](#) tool that appears both in the Visual Studio Installer and in the Visual Studio IDE.
- Suggest a feature, track product issues, and find answers in the [Visual Studio Developer Community](#).
- Use your [GitHub](#) account to talk to us and other Visual Studio developers in the [Visual Studio conversation in the Gitter community](#).

See also

- [Install Visual Studio](#)

- [Disable or move the package cache](#)
- [Use command-line parameters to install Visual Studio](#)

Help Viewer administrator guide

7/11/2019 • 3 minutes to read • [Edit Online](#)

The Help Viewer allows you to manage local Help installations for network environments with or without internet access. Local help content is configured on a per machine basis. By default, users must have administrator rights to update their local Help installation.

If your network environment allows clients to access the internet, you can use the **Help Content Manager** executable to deploy local Help content from the internet. For more information about *HlpCtnMgr.exe* command line syntax, see [Command-line arguments for the Help Content Manager](#).

For information about creating content, creating an intranet service endpoint, and similar types of activities, see the [Help Viewer SDK](#).

If you do not have internet access in your network environment, Help Viewer can deploy local Help content from the intranet or a network share. You can also disable Visual Studio IDE Help options by using [registry key overrides](#) for functionality such as:

- online versus offline help
- content installation at first launch of the IDE
- specifying an intranet content service
- managing content

Deploy local Help content from the internet

You can use **Help Content Manager** (*HlpCtnMgr.exe*) to deploy local Help content from the internet to client computers. Use the following syntax:

```
\%ProgramFiles(x86)%\Microsoft Help Viewer\v2.3\HlpCtnMgr.exe /operation \<*name*> /catalogname \<*catalog name*> /locale \<*locale*>
```

For more information about *HlpCtnMgr.exe* command line syntax, see [Command-line arguments for the Help Content Manager](#).

Requirements:

- Client computers must have access to the internet.
- Users must have administrator rights to update, add, or remove the local Help content after it has been installed.

Caveats:

- The default source for Help will still be online.

Example

The following example installs English content for Visual Studio to a client computer.

To install English content from the internet

1. Choose **Start** and then choose **Run**.
2. Type the following:

```
C:\Program Files (x86)\Microsoft Help Viewer\v2.3\hlpcntmgr.exe /operation install /catalogname  
VisualStudio15 /locale en-us
```

3. Press **Enter**.

Deploy pre-installed local Help content on client computers

You can install a set of content from online to one computer, and then copy that installed set of content to other computers.

Requirements:

- The computer you install the set of content to must have access to the internet.
- Users must have administrator rights to update, add, or remove the local Help content after it has been installed.

TIP

If users do not have administrator rights, it is recommended that you disable the **Manage Content** tab in the Help Viewer. For more information, see [Help Content Manager overrides](#).

Caveats:

- The default source for Help will still be online.

Create the content set

Before you can create the base content set, you must first uninstall all local Visual Studio content on the target computer.

To uninstall local help

1. In the Help Viewer, choose the **Manage Content** tab.
2. Navigate to the Visual Studio document set.
3. Choose **Remove** next to each sub-item.
4. Choose **Update** to uninstall.
5. Browse to `%ProgramData%\Microsoft\HelpLibrary2\Catalogs\VisualStudio15` and verify that the folder only contains the file `catalogType.xml`.

Once you have removed all previously installed local Visual Studio Help content, you are ready to download the base content set.

To download the content

1. In the Help Viewer, choose the **Manage Content** tab.
2. Under **Recommended Documentation** or **Available Documentation**, navigate to the documentation sets you want to download and then choose **Add**.
3. Choose **Update**.

Next, you need to package the content so it can be deployed to client computers.

To package the content

1. Create a folder to copy the content to for later deployment. For example: `C:\VSHelp`.
2. Open `cmd.exe` with Administrator permissions.
3. Navigate to the folder you created in step 1.

4. Type the following:

```
Xcopy %ProgramData%\Microsoft\HelpLibrary2 \<*foldername*>\ /y /e /k /o
```

For example: `Xcopy %ProgramData%\Microsoft\HelpLibrary2 c:\VSHelp\ /y /e /k /o`

Deploy the content

1. Create a network share and copy the help content to that location.

For example, copy the content in `C:\VSHelp` to `\myserver\VSHelp`.

2. Create a `.bat` file to contain the deployment script for the help content. Since the client could possibly have a read lock on any of the files being deleted as part of the push, you should have the client shut down prior to pushing updates. For example:

```
REM - copy pre-ripped content to ProgramData
Xcopy %~dp0HelpLibrary2 %SYSTEMDRIVE%\ProgramData\Microsoft\HelpLibrary2\ /y /e /k /o
if ERRORLEVEL 1 ECHO *** ERROR COPYING Help Library files to ProgramData (%ERRORLEVEL%)
```

3. Run the `.bat` file on the local machines that you want to install the Help content on.

See also

- [Command-line arguments for the Help Content Manager](#)
- [Help Content Manager overrides](#)
- [Microsoft Help Viewer](#)
- [Help Viewer SDK](#)

Command-line arguments for the Help Content Manager

10/18/2019 • 5 minutes to read • [Edit Online](#)

You can specify how to deploy and manage local Help content by using command-line arguments for Help Content Manager (*HlpCtnmgr.exe*). You must run scripts for this command-line tool with administrator permissions, and you can't run these scripts as a service. You can perform the following tasks by using this tool:

- Add or update local Help content from a disk or the cloud.
- Remove local Help content.
- Move the local Help content store.
- Add, update, remove, or move local Help content silently.

Syntax:

```
HlpCtnmgr.exe /operation Value /catalogname CatalogName /locale Locale /sourceuri InstallationPoint
```

For example:

```
hlpctntmgr.exe /operation install /catalogname VisualStudio15 /locale en-us /sourceuri  
d:\productDocumentation\HelpContentSetup.msha
```

NOTE

The catalog name is VisualStudio15 for both Visual Studio 2017 and Visual Studio 2019. This might be unexpected, but this is because the same Help Viewer is used for both Visual Studio versions.

Switches and arguments

The following table defines the switches and arguments that you can use for the command-line tool for Help Content Manager:

SWITCH	REQUIRED?	ARGUMENTS
--------	-----------	-----------

SWITCH	REQUIRED?	ARGUMENTS
/operation	Yes	<ul style="list-style-type: none"> - Install--Adds books from the specified installation source to the local content store. This switch requires the /booklist argument, the /sourceURI argument, or both. If you don't specify the /sourceURI argument, the default Visual Studio URI is used as the installation source. If you don't specify the /booklist argument, all books on the /sourceUri are installed. - Uninstall--Removes the books that you specify from the local content store. This switch requires the /booklist argument or the /sourceURI argument. If you specify the /sourceURI argument, all books are removed, and the /booklist argument is ignored. - Move--Moves the local store to the path that you specify. The default local store path is set as a directory under %ProgramData% This switch requires the /locationPath and /catalogName arguments. Error messages will be logged in the event log if you specify a path that isn't valid or if the drive doesn't contain enough free space to hold the content. - Refresh--Updates topics that have changed since they were installed or most recently updated. This switch requires the /sourceURI argument.
/catalogName	Yes	Specifies the name of the content catalog. For Visual Studio 2017 and Visual Studio 2019, this is VisualStudio15.
/locale	No	<p>Specifies the product locale that's used to view and manage content for the current instance of the Help viewer. For example, you specify EN-US for English-United States.</p> <p>If you don't specify a locale, the locale of the operating system is used. If that locale can't be determined, EN-US is used.</p> <p>If you specify a locale that isn't valid, an error message is logged in the event log.</p>
/e	No	Elevates the Help Content Manager to Administrative privileges if the current user has administrative credentials.

SWITCH	REQUIRED?	ARGUMENTS
/sourceURI	No	<p>Specifies the URL from which content is installed (Service API) or the path to the content installation file (<i>.msha</i>). The URL can point to the Product Group (top-level node) or to the Product Books (leaf-level node) in a Visual Studio 2010 style endpoint. You don't need to include a slash (/) at the end of the URL. If you do include a trailing slash, it will be handled appropriately.</p> <p>An error message is logged in the event log if you specify a file that isn't found, isn't valid, or isn't accessible or if a connection to the internet isn't available or is interrupted while content is being managed.</p>
/vendor	No	<p>Specifies the vendor for the product content that will be removed (for example, Microsoft). The default argument for this switch is Microsoft.</p>
/productName	No	<p>Specifies the product name for the books that will be removed. The product name is identified in the <i>helpcontentsetup.msha</i> or <i>books.html</i> files that shipped with the content. You can remove books from only one product at a time. To remove books from multiple products, you must perform multiple installations.</p>
/booklist	No	<p>Specifies the names of the books to be managed, separated by spaces. Values must match the book names as listed on the installation media.</p> <p>If you don't specify this argument, all recommended books for the specified product in the /sourceURI are installed.</p> <p>If the name of a book contains one or more spaces, surround it with double quotes ("") so that the list is delimited appropriately.</p> <p>Error messages will be logged if you specify a /sourceURI that isn't valid or isn't reachable.</p>
/skuid	No	<p>Specifies the stock keeping unit (SKU) of the product from the installation source, and filters books that the /SourceURI switch identifies.</p>

SWITCH	REQUIRED?	ARGUMENTS
/membership	No	<ul style="list-style-type: none"> - Minimum-- Installs a minimum set of Help content based on the SKU that you specify by using the /skuld switch. The mapping between the SKU and the content set is exposed in the Service API. - Recommended--Installs a set of recommended books for the SKU that you specify by using the /skuld argument. The Installation source is the service API or .MSHA. - Full-- Installs the entire set of books for the SKU that you specify by using the /skuld argument. The Installation source is the service API or .MSHA.
/locationpath	No	Specifies the default folder for local Help content. You must use this switch only to install or move content. If you specify this switch, you must also specify the /silent switch.
/silent	No	Installs or removes Help content without prompting the user or displaying any UI, including the icon in the status notification area. Output is logged to a file in the %Temp% directory. Important: To install content silently, you must use digitally signed .cab files, not .mshc files.
/launchingApp	No	<p>Defines the application and catalog context when the Help viewer is launched without the parent application. The arguments for this switch are <i>CompanyName</i>, <i>ProductName</i>, and <i>VersionNumber</i> (for example,</p> <div style="border: 1px solid black; padding: 5px; display: inline-block;"> <pre>/launchingApp Microsoft,VisualStudio,16.0</pre> </div> <p>).</p> <p>This is required for installing content with the /silent parameter.</p>
/wait Seconds	No	Pauses install, uninstall, and refresh operations. If an operation is already in progress for the catalog, the process will wait up to the given number of seconds to continue. Use 0 to wait indefinitely.
/?	No	Lists the switches and their descriptions for the command-line tool for Help Content Manager.

Exit codes

When you run the command-line tool for the Help Content Manager in silent mode, it returns the following exit codes:

```
Success = 0,  
  
FailureToElevate = 100  
InvalidCmdArgs = 101,  
FailOnFetchingOnlineContent = 110,  
FailOnFetchingContentFromDisk = 120,  
FailOnFetchingInstalledBooks = 130,  
NoBooksToUninstall = 200,  
NoBooksToInstall = 300,  
FailOnUninstall = 400,  
FailOnInstall = 500,  
FailOnMove = 600,  
FailOnUpdate = 700,  
FailOnRefresh = 800,  
Cancelled = 900,  
Others = 999,  
ContentManagementDisabled = 1200,  
OnlineHelpPreferenceDisabled = 1201  
UpdateAlreadyRunning = 1300 - (Signals that the update didn't run because another was in progress.)
```

See also

- [Help Viewer administrator guide](#)
- [Help Content Manager overrides](#)
- [Microsoft Help Viewer](#)

Help Content Manager overrides

7/11/2019 • 2 minutes to read • [Edit Online](#)

You can change the default behavior of Help Viewer and help-related features in the Visual Studio IDE. Some options are specified by creating a [.pkgdef](#) file to set various registry key values. Others are set directly in the registry.

How to control Help Viewer behavior by using a .pkgdef file

1. Create a `.pkgdef` file with the first line as `[$RootKey$\Help]`.
2. Add any or all of the registry key values described in the table below on separate lines, for example
`"UseOnlineHelp"=dword:00000001`.
3. Copy the file to `%ProgramFiles(x86)%\Microsoft Visual Studio\2017\<edition>\Common7\IDE\CommonExtensions`.
4. Run `devenv /updateconfiguration` in a developer command prompt.

Registry key values

REGISTRY KEY VALUE	TYPE	DATA	DESCRIPTION
NewContentAndUpdateService	string	<http URL for service endpoint>	Define a unique service endpoint
UseOnlineHelp	dword	<code>0</code> to specify local Help, <code>1</code> to specify online Help	Define online or offline Help default
OnlineBaseUrl	string	<http URL for service endpoint>	Define a unique F1 endpoint
OnlineHelpPreferenceDisabled	dword	<code>0</code> to enable or <code>1</code> to disable online Help preference option	Disable online Help preference option
DisableManageContent	dword	<code>0</code> to enable or <code>1</code> to disable the Manage Content tab in Help Viewer	Disable the Manage Content tab
DisableFirstRunHelpSelection	dword	<code>0</code> to enable or <code>1</code> to disable help features that are configured the first time that Visual Studio starts	Disable installation of content at first launch of Visual Studio

Example .pkgdef file contents

```

[$RootKey$\Help]
"NewContentAndUpdateService"="https://some.service.endpoint"
"UseOnlineHelp"=dword:00000001
"OnlineBaseUrl"="https://some.service.endpoint"
"OnlineHelpPreferenceDisabled"=dword:00000000
"DisableManageContent"=dword:00000000
"DisableFirstRunHelpSelection"=dword:00000001

```

Use Registry Editor to change Help Viewer behavior

The following two behaviors can be controlled by setting registry key values in the Registry Editor.

TASK	REGISTRY KEY	VALUE	DATA
Override BITS job priority	HKEY_LOCAL_MACHINE\SOFTWARE\Wow6432Node (on a 64-bit machine)\Microsoft\Help\v2.3	BITSPriority	foreground, high, normal, or low
Point to local content store on network share	HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Help\v2.3\Catalogs\VisualStudio15	LocationPath	"ContentStoreNetworkShare"

See also

- [Help Viewer administrator guide](#)
- [Command-line arguments for the Help Content Manager](#)
- [Microsoft Help Viewer](#)

Tools for detecting and managing Visual Studio instances

4/18/2019 • 2 minutes to read • [Edit Online](#)

There are several tools that you can use to detect Visual Studio installations on client machines, and to manage the installations, too.

Detecting existing Visual Studio instances

We have made several tools available that will help you detect and manage installed Visual Studio instances on client machines:

- [vswhere](#): an executable built into Visual Studio or available for separate distribution that helps you find the location of all Visual Studio instances on a particular machine.
- [VS Setup PowerShell](#): PowerShell scripts that use the Setup Configuration API to identify installed instances of Visual Studio.
- [VS-Setup-Samples](#): C# and C++ samples that demonstrate how to use the Setup Configuration API to query an existing installation.

In addition, the [Setup Configuration API](#) provides interfaces for developers who want to build their own utilities for interrogating Visual Studio instances.

Using vswhere.exe

`vswhere.exe` is automatically included in Visual Studio (starting with Visual Studio 2017 version 15.2 and later versions), or you can download it from [the vswhere releases page](#). Use `vswhere -?` to get help information about the tool. As an example, this command shows all releases of Visual Studio, including earlier versions of the product and prereleases, and outputs the results in JSON format:

```
C:\Program Files (x86)\Microsoft Visual Studio\Installer> vswhere.exe -legacy -prerelease -format json
```

TIP

For more information about Visual Studio 2017 installation, see [Visual Studio Setup Archives](#).

Editing the registry for a Visual Studio instance

In Visual Studio, registry settings are stored in a private location, which enables multiple side-by-side instances of the same version of Visual Studio on the same machine.

As these entries are not stored in the global registry, there are special instructions for using the Registry Editor to make changes to registry settings:

1. If you have an open instance of Visual Studio, close it.
2. Start `regedit.exe`.
3. Select the `HKEY_LOCAL_MACHINE` node.

4. From the Regedit main menu, select **File > Load Hive...** and then select the private registry file, which is stored in the **AppData\Local** folder. For example:

```
%localappdata%\Microsoft\VisualStudio\<config>\privateregistry.bin
```

NOTE

`<config>` corresponds to the instance of Visual Studio that you would like to browse.

You will be prompted to provide a hive name, which becomes the name of your isolated hive. After you do so, you should be able to browse the registry under the isolated hive that you created.

IMPORTANT

Before you start Visual Studio again, you must unload the isolated hive that you created. To do this, select **File > Unload Hive** from the Regedit main menu. (If you do not do this, then the file remains locked and Visual Studio will not be able to start.)

Get support

Sometimes, things can go wrong. If your Visual Studio installation fails, see [Troubleshoot Visual Studio installation and upgrade issues](#) for step-by-step guidance.

We also offer a [live chat](#) (English only) support option for installation-related issues.

Here are a few more support options:

- Report product issues to us via the [Report a Problem](#) tool that appears both in the Visual Studio Installer and in the Visual Studio IDE.
- Suggest a feature, track product issues, and find answers in the [Visual Studio Developer Community](#).
- Use your [GitHub](#) account to talk to us and other Visual Studio developers in the [Visual Studio conversation in the Gitter community](#).

See also

- [Visual Studio Administrators Guide](#)

Update a network-based installation of Visual Studio

10/7/2019 • 7 minutes to read • [Edit Online](#)

It's possible to update a network installation layout of Visual Studio with the latest product updates so that it can be used both as an installation point for the latest update of Visual Studio and also to maintain installations that are already deployed to client workstations.

How to update a network layout

To refresh your network install share so that it includes the latest updates, run the `--layout` command to incrementally download updated packages.

New in 15.3: If you selected a partial layout when you first created the network layout, those settings are saved. Any future layout commands use the previous options plus any new options that you specify. But if you are using a layout of an earlier version, you should use the same command-line parameters that you used when you first created the network install layout (in other words, the same workloads and languages) to update its content.

If you selected a partial layout when you first created the network layout, those settings are saved. Any future layout commands use the previous options plus any new options that you specify.

If you host a layout on a file share, you should update a private copy of the layout (for example, `c:\VSLAYOUT`) and then, after all of the updated content is downloaded, copy it to your file share (for example, `\server\products\VS`). If you don't do this, there is a greater chance that any users who run Setup while you are updating the layout might not be able to get all of the content from the layout because it is not yet completely updated.

Let's walk through a few examples of how to create and then update a layout:

- First, here's an example of how to create a layout with one workload for English only:

```
vs_enterprise.exe --layout c:\VSLAYOUT --add Microsoft.VisualStudio.Workload.ManagedDesktop --lang en-US
```

- Here's how to update that same layout to a newer version. You don't have to specify any additional command-line parameters. The previous settings were saved and will be used by any subsequent layout commands in this layout folder.

```
vs_enterprise.exe --layout c:\VSLAYOUT
```

- Here's how to update your layout to a newer version in an unattended manner. The layout operation runs the setup process in a new console window. The window is left open so users can see the final result and a summary of any errors that might have occurred. If you are performing a layout operation in an unattended manner (for example, you have a script that is regularly run to update your layout to the latest version), then use the `--passive` parameter and the process will automatically close the window.

```
vs_enterprise.exe --layout c:\VSLAYOUT --passive
```

- Here's how to add an additional workload and localized language. (This command adds the *Azure development* workload.) Now both Managed Desktop and Azure are included in this layout. The language resources for English and German are also included for all these workloads. And, the layout is

updated to the latest available version.

```
vs_enterprise.exe --layout c:\VSLayout --add Microsoft.VisualStudio.Workload.Azure --lang de-DE
```

IMPORTANT

An update operation doesn't install newly added optional components, even if you include these components in an "add" section of a [response file](#). This occurs because the add operation isn't used during an update.

Workaround: Run a separate modify operation after an upgrade to install the missing components.

- And finally, here's how to add an additional workload and localized language without updating the version. (This command adds the *ASP.NET and web development* workload.) Now the Managed Desktop, Azure, and ASP.NET & Web Development workloads are included in this layout. The language resources for English, German, and French are also included for all these workloads. However, the layout was not updated to the latest available version when this command was run. It remains at the existing version.

```
vs_enterprise.exe --layout c:\VSLayout --add Microsoft.VisualStudio.Workload.NetWeb --lang fr-FR --keepLayoutVersion
```

How to deploy an update to client machines

Depending on how your network environment is configured, an update can either be deployed by an enterprise administrator or initiated from a client machine.

- Users can update a Visual Studio instance that was installed from an offline installation folder:
 - Run the Visual Studio Installer.
 - Then, click **Update**.
- Administrators can update client deployments of Visual Studio without any user interaction with two separate commands:
 - First, update the Visual Studio installer:

```
vs_enterprise.exe --quiet --update
```
 - Then, update the Visual Studio application itself:

```
vs_enterprise.exe update --installPath "C:\Program Files (x86)\Microsoft Visual Studio\2017\Enterprise" --quiet --wait --norestart
```
- Administrators can update client deployments of Visual Studio without any user interaction with two separate commands:
 - First, update the Visual Studio installer:

```
vs_enterprise.exe --quiet --update
```
 - Then, update the Visual Studio application itself:

```
vs_enterprise.exe update --installPath "C:\Program Files (x86)\Microsoft Visual Studio\2019\Enterprise" --quiet --wait --norestart
```

NOTE

Use the [vswhere.exe command](#) to identify the install path of an existing instance of Visual Studio on a client machine.

TIP

For details on how to control when update notifications are presented to users, see [Control updates to network-based Visual Studio deployments](#).

How to verify a layout

Use `--verify` to perform verification on the offline cache supplied. It checks if packages files are either missing or invalid. At the end of the verification, it prints the list of missing files and invalid files.

```
vs_enterprise.exe --layout <layoutDir> --verify
```

The `vs_enterprise.exe` can be invoked inside the `layoutDir`.

NOTE

Some important metadata files that are needed by the `--verify` option must be in the layout offline cache. If these metadata files are missing, "`--verify`" cannot run and Setup gives you an error. If you experience this error, re-create a new offline layout to a different folder (or to the same offline cache folder). To do so, run the same layout command that you used to create the initial offline layout. For example, `vs_enterprise.exe --layout <layoutDir>`.

Microsoft ships updates to Visual Studio periodically, so the new layout that you create might not be the same version as the initial layout.

NOTE

Verification works only for the latest version of a specific minor version of Visual Studio. As soon as a new version is released, verification won't work for earlier patch level releases of the same minor version.

How to fix a layout

Use `--fix` to perform the same verification as `--verify` and also try to fix the identified issues. The `--fix` process needs an internet connection, so make sure your machine is connected to the internet before you invoke `--fix`.

```
vs_enterprise.exe --layout <layoutDir> --fix
```

The `vs_enterprise.exe` can be invoked inside the `layoutDir`.

How to remove older versions from a layout

After you perform layout updates to an offline cache, the layout cache folder may have some obsolete packages that are no longer needed by the latest Visual Studio installation. You can use the `--clean` option to remove obsolete packages from an offline cache folder.

To do this, you'll need the file path(s) to catalog manifest(s) that contain those obsolete packages. You can find the catalog manifests in an "Archive" folder in the offline layout cache. They are saved there when you update a layout. In the "Archive" folder, there is one or more "GUID" named folders, each of which contains an obsolete catalog manifest. The number of "GUID" folders should be the same as the number of updates made to your offline cache.

A few files are saved inside each "GUID" folder. The two files of most interest are a "catalog.json" file and a "version.txt" file. The "catalog.json" file is the obsolete catalog manifest you'll need to pass to the `--clean` option. The other version.txt file contains the version of this obsolete catalog manifest. Based on the version number, you can decide whether you want to remove obsolete packages from this catalog manifest. You can do the same as you go through the other "GUID" folders. After you make the decision on the catalog(s) you want to clean, run the `--clean` command by supplying the files paths to these catalogs.

Here are a few examples of how to use the `--clean` option:

```
vs_enterprise.exe --layout <layoutDir> --clean <file-path-of-catalog1> <file-path-of-catalog2> ...
```

```
vs_enterprise.exe --layout <layoutDir> --clean <file-path-of-catalog1> --clean <file-path-of-catalog2> ...
```

You can also invoke `vs_enterprise.exe` inside the `<layoutDir>`. Here's an example:

```
c:\VSLayout\vs_enterprise.exe --layout c:\VSLayout --clean c:\VSLayout\Archive\1cd70189-fc55-4583-8ad8-a2711e928325\Catalog.json --clean c:\VS2017Layout\Archive\d420889f-6aad-4ba4-99e4-ed7833795a10\Catalog.json
```

When you execute this command, Setup analyzes your offline cache folder to find the list of files that it will remove. You will then have a chance to review the files that are going to be deleted and confirm the deletions.

Get support

Sometimes, things can go wrong. If your Visual Studio installation fails, see [Troubleshoot Visual Studio installation and upgrade issues](#) for step-by-step guidance.

We also offer a [Live chat](#) (English only) support option for installation-related issues.

Here are a few more support options:

- Report product issues to us via the [Report a Problem](#) tool that appears both in the Visual Studio Installer and in the Visual Studio IDE.
- Suggest a feature, track product issues, and find answers in the [Visual Studio Developer Community](#).
- Use your [GitHub](#) account to talk to us and other Visual Studio developers in the [Visual Studio conversation in the Gitter community](#).

See also

- [Install Visual Studio](#)
- [Visual Studio administrator guide](#)
- [Use command-line parameters to install Visual Studio](#)
- [Tools for detecting and managing Visual Studio instances](#)
- [Control updates to network-based Visual Studio deployments](#)
- [Visual Studio product lifecycle and servicing](#)

Update Visual Studio while on a servicing baseline

10/24/2019 • 3 minutes to read • [Edit Online](#)

We update Visual Studio often during its product lifecycle. There are two types of updates:

- **Minor release updates**—for example, 16.0 to 16.1—that include new features and components.
- **Servicing updates**—for example, 16.0.4 to 16.0.5—that include only targeted fixes for critical issues.

Enterprise administrators can choose to keep their clients on a servicing baseline. A servicing baseline is supported with servicing updates for a year past the release of the next servicing baseline.

The servicing baseline option gives developers and administrators more flexibility to adopt the new features, bug fixes, or components included in new minor updates. The first servicing baseline is 16.0.x. For more information, see [Support options for enterprise and professional customers](#).

How to get onto a servicing baseline

To start using a servicing baseline, download a fixed-version Visual Studio installer bootstrapper from [My.VisualStudio.com](#). The bootstrappers have links to the product configurations, workloads, and components for that specific version.

NOTE

Be careful to distinguish between the fixed-version bootstrapper and the standard bootstrappers. The standard bootstrappers are configured to use the latest available release of Visual Studio. The standard bootstrappers have a number in the file name (for example, vs_enterprise_123456789-123456789.exe) when they're downloaded from My.VisualStudio.com.

During installation, enterprise administrators must configure their clients to prevent the clients from updating to the latest release. There are several ways to do this:

- [Change the `channelUri` setting in the response configuration file](#) to use a channel manifest in the layout or local folder.
- [Modify the `channelUri` through command-line execution](#) to use a nonexistent file.
- [Set policies on the client system to disable updates](#), to prevent clients from self-updating.

Install a servicing baseline on a network

Administrators who use a network layout installation should modify the `channelUri` value in the `response.json` file in the layout to use the `channelmanifest.json` file that's in the same folder. For the steps to take, see [Control updates to network-based Visual Studio deployments](#). Changing the `channelUri` value enables clients to look for updates in the layout location.

Install a servicing baseline via the internet

For an internet-based installation, add `--channelUri` with a non-existent channel manifest to the command-line used to launch setup. This disables Visual Studio from using the latest available release for an update. Here's an example:

```
vs_enterprise.exe --channelUri c:\doesnotexist.chman
```

Use policy settings to disable clients from updating

Another option to control updates on a client is to [turn off update notifications](#). Use this option if the channelUri value was not changed on installation. It will disable the client from receiving links to the latest available release. A fixed-version bootstrapper is still necessary to update to a specific version on the client.

How to stay on a servicing baseline

When an update for a servicing baseline is available, fixed-version bootstrapper files are made available for the servicing update at [My.VisualStudio.com](#).

For administrators who deploy by using a network layout install, the administrator should update the [layout location](#). Clients that installed from the location will receive update notifications. If the update must be deployed to clients, follow [these instructions](#). When you modify the 'response.json' for an update, do not add additional workloads, components, or languages. Managing these settings must be done as a 'modify' deployment after the product has been updated.

For an internet-based installation, run the new fixed version bootstrapper with the `--channelUri` parameter pointing to a non-existent channel manifest on the client. If the update is deployed in quiet or passive mode, use two separate commands:

1. Update the Visual Studio installer:

```
vs_enterprise.exe --quiet --update
```

2. Update the Visual Studio application itself:

```
vs_enterprise.exe update --installPath "C:\Program Files (x86)\Microsoft Visual Studio\2019\Enterprise"  
--quiet --wait --norestart --channelUri c:\doesnotexist.chman
```

Get support

Sometimes, things can go wrong. If your Visual Studio installation fails, see [Troubleshoot Visual Studio installation and upgrade issues](#) for step-by-step guidance.

We also offer a [live chat](#) (English only) support option for installation-related issues.

Here are a few more support options:

- Report product issues to us via the [Report a Problem](#) tool that appears both in the Visual Studio Installer and in the Visual Studio IDE.
- Suggest a feature, track product issues, and find answers in the [Visual Studio Developer Community](#).
- Use your [GitHub](#) account to talk to us and other Visual Studio developers in the [Visual Studio conversation in the Gitter community](#).

See also

- [Install Visual Studio](#)
- [Visual Studio administrator guide](#)
- [Use command-line parameters to install Visual Studio](#)
- [Tools for detecting and managing Visual Studio instances](#)
- [How to define settings in a response file](#)
- [Control updates to network-based Visual Studio deployments](#)
- [Visual Studio product lifecycle and servicing](#)

Control updates to network-based Visual Studio deployments

4/18/2019 • 3 minutes to read • [Edit Online](#)

Enterprise administrators often create a layout and host it on a network file share to deploy to their end-users.

Controlling where Visual Studio looks for updates

By default, Visual Studio continues to look online for updates even if the installation was deployed from a network share. If an update is available, the user can install it. Any updated content that is not found in the offline layout is downloaded from the web.

If you want direct control over where Visual Studio looks for updates, you can modify the location where it looks. You can also control the version your users are updated to. To do so, follow these steps:

1. Create an offline layout:

```
vs_enterprise.exe --layout C:\vsoffline --lang en-US
```

2. Copy it to the file share where you want to host it:

```
xcopy /e C:\vsoffline \\server\share\VS
```

3. Modify the response.json file in the layout and change the `channelUri` value to point to a copy of the `channelManifest.json` that the admin controls.

Be sure to escape backslashes in the value, as in the following example:

```
"channelUri": "\\\\"server\\share\\VS\\ChannelManifest.json"
```

Now end-users can run setup from this share to install Visual Studio.

```
\\server\share\VS\vs_enterprise.exe
```

When an enterprise administrator determines it is time for their users to update to a newer version of Visual Studio, they can [update the layout location](#) to incorporate the updated files, as follows.

1. Use a command that is similar to the following command:

```
vs_enterprise.exe --layout \\server\share\VS --lang en-US
```

2. Ensure that the response.json file in the updated layout still contains your customizations, specifically the `channelUri` modification, as follows:

```
"channelUri": "\\\\"server\\share\\VS\\ChannelManifest.json"
```

Existing Visual Studio installs from this layout look for updates at

`\server\share\VS\ChannelManifest.json`. If the channelManifest.json is newer than what the user has installed, Visual Studio notifies the user that an update is available.

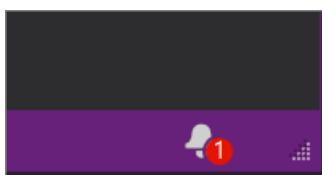
New installs automatically install the updated version of Visual Studio directly from the layout.

Controlling notifications in the Visual Studio IDE

As described earlier, Visual Studio checks the location from which it has been installed, such as a network share or the internet, to see whether any updates are available. When an update is available, Visual Studio notifies the user with a notification flag in the top right-hand corner of the window.



As described earlier, Visual Studio checks the location from which it has been installed, such as a network share or the internet, to see whether any updates are available. When an update is available, Visual Studio notifies the user with a notification icon in the lower right-hand corner of the window.



You can disable the notifications if you don't want end-users to be notified of updates. (For example, you might want to disable notifications if you deliver updates through a central software distribution mechanism.)

Because Visual Studio 2017 [stores registry entries in a private registry](#), you can't directly edit the registry in the typical way. However, Visual Studio includes a `vsregedit.exe` utility that you can use to change Visual Studio settings. You can turn off notifications with the following command:

```
vsregedit.exe set "C:\Program Files (x86)\Microsoft Visual Studio\2017\Enterprise" HKCU ExtensionManager  
AutomaticallyCheckForUpdates2Override dword 0
```

Because Visual Studio 2019 [stores registry entries in a private registry](#), you can't directly edit the registry in the typical way. However, Visual Studio includes a `vsregedit.exe` utility that you can use to change Visual Studio settings. You can turn off notifications with the following command:

```
vsregedit.exe set "C:\Program Files (x86)\Microsoft Visual Studio\2019\Enterprise" HKCU ExtensionManager  
AutomaticallyCheckForUpdates2Override dword 0
```

(Make sure to replace the directory to match the installed instance that you want to edit.)

TIP

Use `vswhere.exe` to find a specific instance of Visual Studio on a client workstation.

Get support

Sometimes, things can go wrong. If your Visual Studio installation fails, see [Troubleshoot Visual Studio installation and upgrade issues](#) for step-by-step guidance.

We also offer a [live chat](#) (English only) support option for installation-related issues.

Here are a few more support options:

- Report product issues to us via the [Report a Problem](#) tool that appears both in the Visual Studio Installer and in the Visual Studio IDE.
- Suggest a feature, track product issues, and find answers in the [Visual Studio Developer Community](#).
- Use your [GitHub](#) account to talk to us and other Visual Studio developers in the [Visual Studio conversation in the Gitter community](#).

See also

- [Install Visual Studio](#)
- [Visual Studio administrator guide](#)
- [Use command-line parameters to install Visual Studio](#)
- [Tools for managing Visual Studio instances](#)
- [Visual Studio product lifecycle and servicing](#)

Package payload changes

5/30/2019 • 2 minutes to read • [Edit Online](#)

Some package payloads are allowed to change after a release has already shipped. When you or someone else creates a layout, this behavior might result in different layout content, depending on when a layout was created.

Verify that a layout includes package payload changes

Here's how to determine if the layout that was previously created has acquired the package payloads that were modified after the release shipped:

1. Open the setup log. The log is typically at `%TEMP%\dd_setup_[date].log` where `[date]` is when the layout operation started in `yyyyMMddHHmmss` format.
2. Look for a line in the log that is structured like the following text:

```
Falling back to signature and signer check because hash verification returned HashMismatch for path:  
[path]
```

3. Then, look for lines later in the log that indicate that the download succeeded for the `[path]`. They might look similar to the following text:

```
Download of [url] succeeded using engine 'WebClient'  
  
END: Downloading [url] to [path]
```

See also

- [Create a network installation of Visual Studio](#)
- [Update a networked-based installation of Visual Studio](#)

Disable or move the package cache

3/25/2019 • 2 minutes to read • [Edit Online](#)

The package cache provides a source of installed packages in case you need to repair Visual Studio or other related products in cases where you have no Internet connection. With some drives or system set ups, however, you might not want to keep all those packages around. The installer will download them when needed, so if you want to save or recover disk space, you can disable or move the package cache.

Disable the package cache

Before you install, modify, or repair Visual Studio or other products with the new installer, you can start the installer with the `--nocache` switch to the installer.

```
"%ProgramFiles(x86)%\Microsoft Visual Studio\Installer\vs_installer.exe" --nocache
```

Any operation you do on any product will remove any existing packages for that product and will avoid saving any packages after they are installed. If you modify or repair Visual Studio and packages are required, they will be downloaded automatically and removed after they are installed.

If you want to re-enable the cache, pass `--cache` instead. Only packages that are required will be cached, so if you need to restore all packages, you should repair Visual Studio before you disconnect from your network.

```
"%ProgramFiles(x86)%\Microsoft Visual Studio\Installer\vs_installer.exe" repair --passive --norestart --cache
```

You can also set the `KeepDownloadedPayloads` [registry policy](#) to disable the cache before you install, modify, or repair Visual Studio.

Move the package cache

A common system configuration is to have Windows installed on an SSD with a larger hard disk (or more) for development needs, such as source code, program binaries, and more. If you want to work offline, you can move the package cache instead.

Currently, you can do this only if you set the `CachePath` [registry policy](#) before you install, modify, or repair Visual Studio.

Get support

Sometimes, things can go wrong. If your Visual Studio installation fails, see [Troubleshoot Visual Studio installation and upgrade issues](#) for step-by-step guidance.

We also offer a [live chat](#) (English only) support option for installation-related issues.

Here are a few more support options:

- Report product issues to us via the [Report a Problem](#) tool that appears both in the Visual Studio Installer and in the Visual Studio IDE.
- Suggest a feature, track product issues, and find answers in the [Visual Studio Developer Community](#).
- Use your [GitHub](#) account to talk to us and other Visual Studio developers in the [Visual Studio conversation in GitHub](#).

[the Gitter community.](#)

See also

- [Install Visual Studio](#)
- [Set defaults for enterprise deployments](#)
- [Use command-line parameters to install Visual Studio](#)

Visual Studio workload and component IDs

9/23/2019 • 2 minutes to read • [Edit Online](#)

Click the edition names in the following table to see the available workload and component IDs you need to install Visual Studio by using a command line, or to specify as a dependency in a VSIX manifest.

Updated for the 15.9 release

EDITION	ID	DESCRIPTION
Visual Studio Enterprise 2017	Microsoft.VisualStudio.Product.Enterprise	Microsoft DevOps solution for productivity and coordination across teams of any size
Visual Studio Professional 2017	Microsoft.VisualStudio.Product.Professional	Professional developer tools and services for small teams
Visual Studio Community 2017	Microsoft.VisualStudio.Product.Community	Free, fully featured IDE for students, open-source, and individual developers
Visual Studio Team Explorer 2017	Microsoft.VisualStudio.Product.TeamExplorer	Interact with Team Foundation Server and Azure DevOps Services without a Visual Studio developer toolset
Visual Studio Desktop Express 2017	Microsoft.VisualStudio.Product.WDExpress	Build Native and Managed applications like WPF, WinForms, and Win32 with syntax-aware code editing, source code control, and work item management. Includes support for C#, Visual Basic, and Visual C++.
Visual Studio Build Tools 2017	Microsoft.VisualStudio.Product.BuildTools	The Visual Studio Build Tools allows you to build native and managed MSBuild-based applications without requiring the Visual Studio IDE. There are options to install the Visual C++ compilers and libraries, MFC, ATL, and C++/CLI support.
Visual Studio Test Agent 2017	Microsoft.VisualStudio.Product.TestAgent	Supports running automated tests and load tests remotely
Visual Studio Test Controller 2017	Microsoft.VisualStudio.Product.TestController	Distribute automated tests to multiple machines
Visual Studio Test Professional 2017	Microsoft.VisualStudio.Product.TestProfessional	Visual Studio Test Professional 2017
Visual Studio Feedback Client 2017	Microsoft.VisualStudio.Product.FeedbackClient	Visual Studio Feedback Client 2017

For more information about how to use these lists, see the [Use command-line parameters to install Visual Studio 2017](#) page and the [How to: Migrate extensibility projects to Visual Studio 2017](#) page.

Updated for the 16.3 release

EDITION	ID	DESCRIPTION
Visual Studio Enterprise 2019	Microsoft.VisualStudio.Product.Enterprise	Microsoft DevOps solution for productivity and coordination across teams of any size
Visual Studio Professional 2019	Microsoft.VisualStudio.Product.Professional	Professional developer tools and services for small teams
Visual Studio Community 2019	Microsoft.VisualStudio.Product.Community	Free, fully featured IDE for students, open-source, and individual developers
Visual Studio Team Explorer 2019	Microsoft.VisualStudio.Product.TeamExplorer	Interact with Team Foundation Server and Azure DevOps Services without a Visual Studio developer toolset
Visual Studio Build Tools 2019	Microsoft.VisualStudio.Product.BuildTools	The Visual Studio Build Tools allows you to build native and managed MSBuild-based applications without requiring the Visual Studio IDE. There are options to install the Visual C++ compilers and libraries, MFC, ATL, and C++/CLI support.
Visual Studio Test Agent 2019	Microsoft.VisualStudio.Product.TestAgent	Supports running automated tests and load tests remotely
Visual Studio Load Test Controller 2019	Microsoft.VisualStudio.Product.TestController	Distribute automated tests to multiple machines

For more information about how to use these lists, see the [Use command-line parameters to install Visual Studio](#) page and the [How to: Migrate extensibility projects to Visual Studio](#) page.

NOTE

For a list of the workload and component IDs for the previous version, see [Visual Studio 2017 workload and component IDs](#)

Get support

Sometimes, things can go wrong. If your Visual Studio installation fails, see [Troubleshoot Visual Studio installation and upgrade issues](#) for step-by-step guidance.

We also offer a [live chat](#) (English only) support option for installation-related issues.

Here are a few more support options:

- Report product issues to us via the [Report a Problem](#) tool that appears both in the Visual Studio Installer and in the Visual Studio IDE.
- Suggest a feature, track product issues, and find answers in the [Visual Studio Developer Community](#).

- Use your [GitHub](#) account to talk to us and other Visual Studio developers in the [Visual Studio conversation in the Gitter community](#).

See also

- [Visual Studio administrator guide for Visual Studio](#)
- [Use command-line parameters to install Visual Studio](#)
 - [Command-line parameter examples](#)
- [Create an offline installation of Visual Studio](#)

Visual Studio Enterprise component directory

12/4/2019 • 70 minutes to read • [Edit Online](#)

The tables on this page list the IDs that you can use to install Visual Studio from the command line, or that you can specify as a dependency in a VSIX manifest. We'll add additional components when we release updates to Visual Studio.

Also note the following about the page:

- Each workload has its own section, followed by the workload ID and a table of the components that are available for the workload.
- By default, the **Required** components will be installed when you install the workload.
- If you choose to, you can also install the **Recommended** and **Optional** components.
- We've also added a section that lists the additional components that are not affiliated with any workload.

When you set dependencies in your VSIX manifest, you must specify Component IDs only. Use the tables on this page to determine the minimum component dependencies. In some scenarios, this might mean that you specify only one component from a workload. In other scenarios, it might mean that you specify multiple components from a single workload or multiple components from multiple workloads. For more information, see the [How to: Migrate extensibility projects to Visual Studio 2017](#) page.

For more information about how to use these IDs, see the [Use command-line parameters to install Visual Studio 2017](#) page. And, for a list of workload and component IDs for other products, see the [Visual Studio workload and component IDs](#) page.

Visual Studio core editor (included with Visual Studio Enterprise 2017)

ID: Microsoft.VisualStudio.Workload.CoreEditor

Description: The Visual Studio core shell experience, including syntax-aware code editing, source code control and work item management.

Components included by this workload

COMPONENT ID	NAME	VERSION	DEPENDENCY TYPE
Microsoft.VisualStudio.Component.CoreEditor	Visual Studio core editor	15.8.27729.1	Required
Microsoft.VisualStudio.Component.StartPageExperiment.Cpp	Visual Studio Start Page for C++ Users	15.0.27128.1	Optional

Azure development

ID: Microsoft.VisualStudio.Workload.Azure

Description: Azure SDKs, tools, and projects for developing cloud apps, creating resources, and building Containers including Docker support.

Components included by this workload

Component ID	Name	Version	Dependency Type
Component.Microsoft.VisualStudio.RazorExtension	Razor Language Services	15.0.26720.2	Required
Component.Microsoft.VisualStudio.Web.AzureFunctions	Microsoft Azure WebJobs Tools	15.7.27617.1	Required
Component.Microsoft.Web.LibraryManager	Library Manager	15.8.27705.0	Required
Component.WebSocket	WebSocket4Net	15.0.26606.0	Required
Microsoft.Component.ClickOnce	ClickOnce Publishing	15.8.27825.0	Required
Microsoft.Component.MSBuild	MSBuild	15.7.27520.0	Required
Microsoft.Component.NetFX.Core.Runtime	.NET Core runtime	15.0.26208.0	Required
Microsoft.Net.Component.4.5.2.TargetingPack	.NET Framework 4.5.2 targeting pack	15.6.27406.0	Required
Microsoft.Net.Component.4.5.TargetingPack	.NET Framework 4.5 targeting pack	15.6.27406.0	Required
Microsoft.Net.Component.4.6.1.SDK	.NET Framework 4.6.1 SDK	15.6.27406.0	Required
Microsoft.Net.Component.4.6.1.TargetingPack	.NET Framework 4.6.1 targeting pack	15.6.27406.0	Required
Microsoft.Net.ComponentGroup.DevelopmentPrerequisites	.NET Framework 4.6.1 development tools	15.8.27825.0	Required
Microsoft.Net.Core.Component.SDK.2.1	.NET Core 2.1 development tools	15.8.27924.0	Required
Microsoft.NetCore.ComponentGroup.DevelopmentTools.2.1	.NET Core 2.1 development tools	15.8.27924.0	Required
Microsoft.NetCore.ComponentGroup.Web.2.1	.NET Core 2.1 development tools	15.8.27924.0	Required
Microsoft.VisualStudio.Component.Azure.AuthoringTools	Azure Authoring Tools	15.9.28307.421	Required
Microsoft.VisualStudio.Component.Azure.ClientLibs	Azure libraries for .NET	15.0.26208.0	Required

Component ID	Name	Version	Dependency Type
Microsoft.VisualStudio.Component.Azure.Compute Emulator	Azure Compute Emulator	15.9.28307.421	Required
Microsoft.VisualStudio.Component.Azure.Storage.Emulator	Azure Storage Emulator	15.9.28125.51	Required
Microsoft.VisualStudio.Component.CloudExplorer	Cloud Explorer	15.9.28230.55	Required
Microsoft.VisualStudio.Component.Common.Azure.Tools	Connectivity and publishing tools	15.9.28107.0	Required
Microsoft.VisualStudio.Component.DockerTools	Container development tools	15.8.27906.1	Required
Microsoft.VisualStudio.Component.DockerTools.BuildTools	Container development tools - Build Tools	15.7.27617.1	Required
Microsoft.VisualStudio.Component.FSharp	F# language support	15.8.27825.0	Required
Microsoft.VisualStudio.Component.FSharp.WebTemplates	F# language support for web projects	15.9.28307.421	Required
Microsoft.VisualStudio.Component.IISExpress	IIS Express	15.0.26208.0	Required
Microsoft.VisualStudio.Component.JavaScript.Diagnostics	JavaScript diagnostics	15.8.27729.1	Required
Microsoft.VisualStudio.Component.JavaScript.TypeScript	JavaScript and TypeScript language support	15.9.28125.51	Required
Microsoft.VisualStudio.Component.ManagedDesktop.Core	Managed Desktop Workload Core	15.8.27729.1	Required
Microsoft.VisualStudio.Component.NuGet	NuGet package manager	15.9.28016.0	Required
Microsoft.VisualStudio.Component.PortableLibrary	.NET Portable Library targeting pack	15.6.27309.0	Required
Microsoft.VisualStudio.Component.Roslyn.Compiler	C# and Visual Basic Roslyn compilers	15.6.27309.0	Required
Microsoft.VisualStudio.Component.Roslyn.LanguageServices	C# and Visual Basic	15.8.27729.1	Required

Component ID	Name	Version	Dependency Type
Microsoft.VisualStudio.Component.SQL.ADAL	SQL ADAL runtime	15.6.27406.0	Required
Microsoft.VisualStudio.Component.SQL.CLR	CLR data types for SQL Server	15.0.26208.0	Required
Microsoft.VisualStudio.Component.SQL.CMDUtils	SQL Server Command Line Utilities	15.0.26208.0	Required
Microsoft.VisualStudio.Component.SQL.DataSources	Data sources for SQL Server support	15.0.26621.2	Required
Microsoft.VisualStudio.Component.SQL.LocalDB.Runtime	SQL Server Express 2016 LocalDB	15.7.27617.1	Required
Microsoft.VisualStudio.Component.SQL.NCLI	SQL Server Native Client	15.0.26208.0	Required
Microsoft.VisualStudio.Component.SQL.SSDT	SQL Server Data Tools	15.9.28107.0	Required
Microsoft.VisualStudio.Component.Static.Analysis.Tools	Static analysis tools	15.0.26208.0	Required
Microsoft.VisualStudio.Component.TextTemplating	Text Template Transformation	15.0.26208.0	Required
Microsoft.VisualStudio.Component.TypeScript.3.1	TypeScript 3.1 SDK	15.0.28218.60	Required
Microsoft.VisualStudio.Component.VisualStudioData	Data sources and service references	15.6.27406.0	Required
Microsoft.VisualStudio.Component.Web	ASP.NET and web development tools	15.8.27825.0	Required
Microsoft.VisualStudio.ComponentGroup.Azure.Prerequisites	Azure development prerequisites	15.9.28107.0	Required
Microsoft.VisualStudio.ComponentGroup.AzureFunctions	Microsoft Azure WebJobs Tools	15.7.27617.1	Required
Microsoft.VisualStudio.ComponentGroup.Web	ASP.NET and web development tools prerequisites	15.9.28219.51	Required
Microsoft.VisualStudio.ComponentGroup.WebToolsExtensions	ASP.NET and web development	15.8.27825.0	Required
Microsoft.Component.Azure.DataLake.Tools	Azure Data Lake and Stream Analytics Tools	15.9.28107.0	Recommended

Component ID	Name	Version	Dependency Type
Microsoft.Net.Component.4.5.1.TargetingPack	.NET Framework 4.5.1 targeting pack	15.6.27406.0	Recommended
Microsoft.Net.Component.4.6.TargetingPack	.NET Framework 4.6 targeting pack	15.6.27406.0	Recommended
Microsoft.Net.Component.4.TargetingPack	.NET Framework 4 targeting pack	15.6.27406.0	Recommended
Microsoft.ComponentGroup.TargetingPacks.Common	.NET Framework 4 – 4.6 development tools	15.6.27406.0	Recommended
Microsoft.VisualStudio.Component.AspNet45	Advanced ASP.NET features	15.7.27625.0	Recommended
Microsoft.VisualStudio.Component.Azure.MobileAppsSdk	Azure Mobile Apps SDK	15.7.27625.0	Recommended
Microsoft.VisualStudio.Component.Azure.ResourceManager.Tools	Azure Resource Manager core tools	15.9.28107.0	Recommended
Microsoft.VisualStudio.Component.Azure.ServiceFabric.Tools	Service Fabric Tools	15.8.27825.0	Recommended
Microsoft.VisualStudio.Component.Azure.Waverton	Azure Cloud Services core tools	15.9.28107.0	Recommended
Microsoft.VisualStudio.Component.Azure.Waverton.BuildTools	Azure Cloud Services build tools	15.7.27617.1	Recommended
Microsoft.VisualStudio.Component.Debugger.Snapshot	Snapshot Debugger	15.8.28010.0	Recommended
Microsoft.VisualStudio.Component.DiagnosticTools	.NET profiling tools	15.8.27729.1	Recommended
Microsoft.VisualStudio.Component.IntelliTrace.FrontEnd	IntelliTrace	15.8.27729.1	Recommended
Microsoft.VisualStudio.Component.WebDeploy	Web Deploy	15.8.27729.1	Recommended
Microsoft.VisualStudio.ComponentGroup.Azure.CloudServices	Azure Cloud Services tools	15.0.26504.0	Recommended
Microsoft.VisualStudio.ComponentGroup.Azure.ResourceManager.Tools	Azure Resource Manager tools	15.0.27005.2	Recommended

Component ID	Name	Version	Dependency Type
Microsoft.Net.Component.4.6.2.SDK	.NET Framework 4.6.2 SDK	15.6.27406.0	Optional
Microsoft.Net.Component.4.6.2.TargetingPack	.NET Framework 4.6.2 targeting pack	15.6.27406.0	Optional
Microsoft.Net.Component.4.7.1.SDK	.NET Framework 4.7.1 SDK	15.6.27406.0	Optional
Microsoft.Net.Component.4.7.1.TargetingPack	.NET Framework 4.7.1 targeting pack	15.6.27406.0	Optional
Microsoft.Net.Component.4.7.2.SDK	.NET Framework 4.7.2 SDK	15.8.27825.0	Optional
Microsoft.Net.Component.4.7.2.TargetingPack	.NET Framework 4.7.2 targeting pack	15.8.27825.0	Optional
Microsoft.Net.Component.4.7.SDK	.NET Framework 4.7 SDK	15.6.27406.0	Optional
Microsoft.Net.Component.4.7.TargetingPack	.NET Framework 4.7 targeting pack	15.6.27406.0	Optional
Microsoft.Net.ComponentGroup.4.6.2.DeveloperTools	.NET Framework 4.6.2 development tools	15.6.27406.0	Optional
Microsoft.Net.ComponentGroup.4.7.1.DeveloperTools	.NET Framework 4.7.1 development tools	15.6.27406.0	Optional
Microsoft.Net.ComponentGroup.4.7.2.DeveloperTools	.NET Framework 4.7.2 development tools	15.8.27825.0	Optional
Microsoft.Net.ComponentGroup.4.7.DeveloperTools	.NET Framework 4.7 development tools	15.6.27406.0	Optional
Microsoft.Net.Core.Component.SDK	.NET Core 2.0 development tools	15.6.27406.0	Optional
Microsoft.Net.Core.Component.SDK.1x	.NET Core 1.0 - 1.1 development tools	15.6.27406.0	Optional
Microsoft.NetCore.1x.ComponentGroup.Web	.NET Core 1.0 - 1.1 development tools for Web	15.6.27406.0	Optional
Microsoft.NetCore.ComponentGroup.DevelopmentTools	.NET Core 2.0 development tools	15.8.27729.1	Optional
Microsoft.NetCore.ComponentGroup.Web	.NET Core 2.0 development tools	15.7.27625.0	Optional
Microsoft.VisualStudio.Component.Azure.Storage.AzCopy	Azure Storage AzCopy	15.0.26906.1	Optional

Component ID	Name	Version	Dependency Type
Microsoft.VisualStudio.Component.Wcf.Tooling	Windows Communication Foundation	15.8.27924.0	Optional

Data storage and processing

ID: Microsoft.VisualStudio.Workload.Data

Description: Connect, develop, and test data solutions with SQL Server, Azure Data Lake, or Hadoop.

Components included by this workload

Component ID	Name	Version	Dependency Type
Component.Microsoft.VisualStudio.RazorExtension	Razor Language Services	15.0.26720.2	Recommended
Component.Microsoft.Web.LibraryManager	Library Manager	15.8.27705.0	Recommended
Component.Redgate.ReadyRoll	Redgate ReadyRoll Core	1.17.18155.10346	Recommended
Component.Redgate.SQLPrompt.VsPackage	Redgate SQL Prompt Core	9.2.0.5601	Recommended
Component.Redgate.SQLSearch.VSExtension	Redgate SQL Search	3.1.7.2062	Recommended
Component.WebSocket	WebSocket4Net	15.0.26606.0	Recommended
Microsoft.Component.Azure.DataLake.Tools	Azure Data Lake and Stream Analytics Tools	15.9.28107.0	Recommended
Microsoft.Component.ClickOnce	ClickOnce Publishing	15.8.27825.0	Recommended
Microsoft.Component_MSBuild	MSBuild	15.7.27520.0	Recommended
Microsoft.Net.Component.4.5.1.TargetingPack	.NET Framework 4.5.1 targeting pack	15.6.27406.0	Recommended
Microsoft.Net.Component.4.5.2.TargetingPack	.NET Framework 4.5.2 targeting pack	15.6.27406.0	Recommended
Microsoft.Net.Component.4.5.TargetingPack	.NET Framework 4.5 targeting pack	15.6.27406.0	Recommended
Microsoft.Net.Component.4.6.1.SDK	.NET Framework 4.6.1 SDK	15.6.27406.0	Recommended
Microsoft.Net.Component.4.6.1.TargetingPack	.NET Framework 4.6.1 targeting pack	15.6.27406.0	Recommended

Component ID	Name	Version	Dependency Type
Microsoft.Net.Component.4.6.TargetingPack	.NET Framework 4.6 targeting pack	15.6.27406.0	Recommended
Microsoft.Net.Component.4.TargetingPack	.NET Framework 4 targeting pack	15.6.27406.0	Recommended
Microsoft.Net.ComponentGroup.DevelopmentPrerequisites	.NET Framework 4.6.1 development tools	15.8.27825.0	Recommended
Microsoft.Net.ComponentGroup.TargetingPacks.Common	.NET Framework 4 – 4.6 development tools	15.6.27406.0	Recommended
Microsoft.Net.Core.Component.SDK.2.1	.NET Core 2.1 development tools	15.8.27924.0	Recommended
Microsoft.VisualStudio.Component.Azure.AuthoringTools	Azure Authoring Tools	15.9.28307.421	Recommended
Microsoft.VisualStudio.Component.Azure.ClientLibs	Azure libraries for .NET	15.0.26208.0	Recommended
Microsoft.VisualStudio.Component.Azure.Compute.Emulator	Azure Compute Emulator	15.9.28307.421	Recommended
Microsoft.VisualStudio.Component.Azure.Storage.Emulator	Azure Storage Emulator	15.9.28125.51	Recommended
Microsoft.VisualStudio.Component.Azure.Waverton	Azure Cloud Services core tools	15.9.28107.0	Recommended
Microsoft.VisualStudio.Component.Azure.Waverton.BuildTools	Azure Cloud Services build tools	15.7.27617.1	Recommended
Microsoft.VisualStudio.Component.CloudExplorer	Cloud Explorer	15.9.28230.55	Recommended
Microsoft.VisualStudio.Component.Common.Azure.Tools	Connectivity and publishing tools	15.9.28107.0	Recommended
Microsoft.VisualStudio.Component.DockerTools	Container development tools	15.8.27906.1	Recommended
Microsoft.VisualStudio.Component.DockerTools.BuildTools	Container development tools - Build Tools	15.7.27617.1	Recommended
Microsoft.VisualStudio.Component.IISExpress	IIS Express	15.0.26208.0	Recommended

Component ID	Name	Version	Dependency Type
Microsoft.VisualStudio.Component.JavaScript.Diagnostics	JavaScript diagnostics	15.8.27729.1	Recommended
Microsoft.VisualStudio.Component.JavaScript.TypeScript	JavaScript and TypeScript language support	15.9.28125.51	Recommended
Microsoft.VisualStudio.Component.ManagedDesktop.Core	Managed Desktop Workload Core	15.8.27729.1	Recommended
Microsoft.VisualStudio.Component.NuGet	NuGet package manager	15.9.28016.0	Recommended
Microsoft.VisualStudio.Component.PortableLibrary	.NET Portable Library targeting pack	15.6.27309.0	Recommended
Microsoft.VisualStudio.Component.Roslyn.Compiler	C# and Visual Basic Roslyn compilers	15.6.27309.0	Recommended
Microsoft.VisualStudio.Component.Roslyn.LanguageServices	C# and Visual Basic	15.8.27729.1	Recommended
Microsoft.VisualStudio.Component.SQL.ADAL	SQL ADAL runtime	15.6.27406.0	Recommended
Microsoft.VisualStudio.Component.SQL.CLR	CLR data types for SQL Server	15.0.26208.0	Recommended
Microsoft.VisualStudio.Component.SQL.CMDUtils	SQL Server Command Line Utilities	15.0.26208.0	Recommended
Microsoft.VisualStudio.Component.SQL.DataSources	Data sources for SQL Server support	15.0.26621.2	Recommended
Microsoft.VisualStudio.Component.SQL.LocalDB.Runtime	SQL Server Express 2016 LocalDB	15.7.27617.1	Recommended
Microsoft.VisualStudio.Component.SQL.NCLI	SQL Server Native Client	15.0.26208.0	Recommended
Microsoft.VisualStudio.Component.SQL.SSDT	SQL Server Data Tools	15.9.28107.0	Recommended
Microsoft.VisualStudio.Component.Static.Analysis.Tools	Static analysis tools	15.0.26208.0	Recommended
Microsoft.VisualStudio.Component.TextTemplating	Text Template Transformation	15.0.26208.0	Recommended
Microsoft.VisualStudio.Component.TypeScript.3.1	TypeScript 3.1 SDK	15.0.28218.60	Recommended

Component ID	Name	Version	Dependency Type
Microsoft.VisualStudio.Component.VisualStudioData	Data sources and service references	15.6.27406.0	Recommended
Microsoft.VisualStudio.Component.Web	ASP.NET and web development tools	15.8.27825.0	Recommended
Microsoft.VisualStudio.ComponentGroup.Web	ASP.NET and web development tools prerequisites	15.9.28219.51	Recommended
Microsoft.VisualStudio.ComponentGroup.WebToolsExtensions	ASP.NET and web development	15.8.27825.0	Recommended
Microsoft.VisualStudio.Component.FSharp.Desktop	F# desktop language support	15.8.27825.0	Optional

Data science and analytical applications

ID: Microsoft.VisualStudio.Workload.DataScience

Description: Languages and tooling for creating data science applications, including Python, R and F#.

Components included by this workload

Component ID	Name	Version	Dependency Type
Component.Anconda3.x64	Anaconda3 64-bit (5.2.0)	5.2.0	Recommended
Microsoft.Component.CookiecutterTools	Cookiecutter template support	15.0.26621.2	Recommended
Microsoft.Component.PythonTools	Python language support	15.0.26823.1	Recommended
Microsoft.Component.PythonTools.Web	Python web support	15.9.28107.0	Recommended
Microsoft.Net.Component.4.6.1.TargetingPack	.NET Framework 4.6.1 targeting pack	15.6.27406.0	Recommended
Microsoft.VisualStudio.Component.Common.Azure.Tools	Connectivity and publishing tools	15.9.28107.0	Recommended
Microsoft.VisualStudio.Component.FSharp.Desktop	F# desktop language support	15.8.27825.0	Recommended
Microsoft.VisualStudio.Component.JavaScript.TypeScript	JavaScript and TypeScript language support	15.9.28125.51	Recommended
Microsoft.VisualStudio.Component.NuGet	NuGet package manager	15.9.28016.0	Recommended

Component ID	Name	Version	Dependency Type
Microsoft.VisualStudio.Component.R.Open	Microsoft R Client (3.3.2)	15.6.27406.0	Recommended
Microsoft.VisualStudio.Component.RHost	Runtime support for R development tools	15.6.27406.0	Recommended
Microsoft.VisualStudio.Component.Roslyn.Compiler	C# and Visual Basic Roslyn compilers	15.6.27309.0	Recommended
Microsoft.VisualStudio.Component.Roslyn.LanguageServices	C# and Visual Basic	15.8.27729.1	Recommended
Microsoft.VisualStudio.Component.RTools	R language support	15.0.26919.1	Recommended
Microsoft.VisualStudio.Component.SQL.CLR	CLR data types for SQL Server	15.0.26208.0	Recommended
Microsoft.VisualStudio.Component.Static.Analysis.Tools	Static analysis tools	15.0.26208.0	Recommended
Microsoft.VisualStudio.Component.TypeScript.3.1	TypeScript 3.1 SDK	15.0.28218.60	Recommended
Microsoft.VisualStudio.Component.VisualStudioData	Data sources and service references	15.6.27406.0	Recommended
Microsoft.VisualStudio.Component.WebDeploy	Web Deploy	15.8.27729.1	Recommended
Microsoft.VisualStudio.ComponentGroup.WebToolsExtensions	ASP.NET and web development	15.8.27825.0	Recommended
Component.Anaconda2.x64	Anaconda2 64-bit (5.2.0)	5.2.0	Optional
Component.Anaconda2.x86	Anaconda2 32-bit (5.2.0)	5.2.0	Optional
Component.Anaconda3.x86	Anaconda3 32-bit (5.2.0)	5.2.0	Optional
Microsoft.Component.VC.Runtime.UCRTSDK	Windows Universal CRT SDK	15.6.27309.0	Optional
Microsoft.ComponentGroup.PythonTools.NativeDevelopment	Python native development tools	15.9.28307.102	Optional
Microsoft.VisualStudio.Component.Graphics.Tools	Graphics debugger and GPU profiler for DirectX	15.6.27406.0	Optional
Microsoft.VisualStudio.Component.Graphics.Win81	Graphics Tools Windows 8.1 SDK	15.6.27406.0	Optional

Component ID	Name	Version	Dependency Type
Microsoft.VisualStudio.Component.VC.140	VC++ 2015.3 v14.00 (v140) toolset for desktop	15.7.27617.1	Optional
Microsoft.VisualStudio.Component.VC.CoreRedist	Visual Studio C++ core features	15.6.27406.0	Optional
Microsoft.VisualStudio.Component.VC.DiagnosticTools	C++ profiling tools	15.0.26823.1	Optional
Microsoft.VisualStudio.Component.VC.Tools.x86.x64	VC++ 2017 version 15.9 v14.16 latest v141 tools	15.9.28230.55	Optional
Microsoft.VisualStudio.Component.Windows10SDK	Windows Universal C Runtime	15.6.27406.0	Optional
Microsoft.VisualStudio.Component.Windows10SDK.17763	Windows 10 SDK (10.0.17763.0)	15.9.28307.102	Optional
Microsoft.VisualStudio.Component.Windows81SDK	Windows 8.1 SDK	15.6.27406.0	Optional

.NET desktop development

ID: Microsoft.VisualStudio.Workload.ManagedDesktop

Description: Build WPF, Windows Forms, and console applications using C#, Visual Basic, and F#.

Components included by this workload

Component ID	Name	Version	Dependency Type
Microsoft.Component.ClickOnce	ClickOnce Publishing	15.8.27825.0	Required
Microsoft.Component.MSBuild	MSBuild	15.7.27520.0	Required
Microsoft.Net.Component.4.6.1.SDK	.NET Framework 4.6.1 SDK	15.6.27406.0	Required
Microsoft.Net.Component.4.6.1.TargetingPack	.NET Framework 4.6.1 targeting pack	15.6.27406.0	Required
Microsoft.Net.ComponentGroup.DevelopmentPrerequisites	.NET Framework 4.6.1 development tools	15.8.27825.0	Required
Microsoft.VisualStudio.Component.ManagedDesktop.Core	Managed Desktop Workload Core	15.8.27729.1	Required

Component ID	Name	Version	Dependency Type
Microsoft.VisualStudio.Component.ManagedDesktop.Prerequisites	.NET desktop development tools	15.7.27625.0	Required
Microsoft.VisualStudio.Component.PortableLibrary	.NET Portable Library targeting pack	15.6.27309.0	Required
Microsoft.VisualStudio.Component.Roslyn.Compiler	C# and Visual Basic Roslyn compilers	15.6.27309.0	Required
Microsoft.VisualStudio.Component.Roslyn.LanguageServices	C# and Visual Basic	15.8.27729.1	Required
Microsoft.VisualStudio.Component.SQL.CLR	CLR data types for SQL Server	15.0.26208.0	Required
Microsoft.VisualStudio.Component.Static.Analysis.Tools	Static analysis tools	15.0.26208.0	Required
Microsoft.VisualStudio.Component.TextTemplating	Text Template Transformation	15.0.26208.0	Required
Microsoft.VisualStudio.Component.VisualStudioData	Data sources and service references	15.6.27406.0	Required
Microsoft.ComponentGroup.Blend	Blend for Visual Studio	15.6.27406.0	Recommended
Microsoft.Net.Component.4.5.1.TargetingPack	.NET Framework 4.5.1 targeting pack	15.6.27406.0	Recommended
Microsoft.Net.Component.4.5.2.TargetingPack	.NET Framework 4.5.2 targeting pack	15.6.27406.0	Recommended
Microsoft.Net.Component.4.5.TargetingPack	.NET Framework 4.5 targeting pack	15.6.27406.0	Recommended
Microsoft.Net.Component.4.6.TargetingPack	.NET Framework 4.6 targeting pack	15.6.27406.0	Recommended
Microsoft.Net.Component.4.TargetingPack	.NET Framework 4 targeting pack	15.6.27406.0	Recommended
Microsoft.Net.ComponentGroup.TargetingPacks.Common	.NET Framework 4 – 4.6 development tools	15.6.27406.0	Recommended
Microsoft.VisualStudio.Component.Debugger.JustInTime	Just-In-Time debugger	15.0.27005.2	Recommended
Microsoft.VisualStudio.Component.DiagnosticTools	.NET profiling tools	15.8.27729.1	Recommended

Component ID	Name	Version	Dependency Type
Microsoft.VisualStudio.Component.EntityFramework	Entity Framework 6 tools	15.6.27406.0	Recommended
Microsoft.VisualStudio.Component.IntelliTrace.FrontEnd	IntelliTrace	15.8.27729.1	Recommended
Microsoft.VisualStudio.Component.LiveUnitTesting	Live Unit Testing	15.0.26720.2	Recommended
Component.Dotfuscator	PreEmptive Protection - Dotfuscator	15.0.26208.0	Optional
Component.Microsoft.VisualStudio.RazorExtension	Razor Language Services	15.0.26720.2	Optional
Component.Microsoft.Web.LibraryManager	Library Manager	15.8.27705.0	Optional
Component.WebSocket	WebSocket4Net	15.0.26606.0	Optional
Microsoft.Net.Component.4.6.2.SDK	.NET Framework 4.6.2 SDK	15.6.27406.0	Optional
Microsoft.Net.Component.4.6.2.TargetingPack	.NET Framework 4.6.2 targeting pack	15.6.27406.0	Optional
Microsoft.Net.Component.4.7.1.SDK	.NET Framework 4.7.1 SDK	15.6.27406.0	Optional
Microsoft.Net.Component.4.7.1.TargetingPack	.NET Framework 4.7.1 targeting pack	15.6.27406.0	Optional
Microsoft.Net.Component.4.7.2.SDK	.NET Framework 4.7.2 SDK	15.8.27825.0	Optional
Microsoft.Net.Component.4.7.2.TargetingPack	.NET Framework 4.7.2 targeting pack	15.8.27825.0	Optional
Microsoft.Net.Component.4.7.SDK	.NET Framework 4.7 SDK	15.6.27406.0	Optional
Microsoft.Net.Component.4.7.TargetingPack	.NET Framework 4.7 targeting pack	15.6.27406.0	Optional
Microsoft.Net.ComponentGroup.4.6.2.DeveloperTools	.NET Framework 4.6.2 development tools	15.6.27406.0	Optional
Microsoft.Net.ComponentGroup.4.7.1.DeveloperTools	.NET Framework 4.7.1 development tools	15.6.27406.0	Optional
Microsoft.Net.ComponentGroup.4.7.2.DeveloperTools	.NET Framework 4.7.2 development tools	15.8.27825.0	Optional

Component ID	Name	Version	Dependency Type
Microsoft.Net.ComponentGroup.4.7.DeveloperTools	.NET Framework 4.7 development tools	15.6.27406.0	Optional
Microsoft.Net.Core.Component.SDK	.NET Core 2.0 development tools	15.6.27406.0	Optional
Microsoft.Net.Core.Component.SDK.1x	.NET Core 1.0 - 1.1 development tools	15.6.27406.0	Optional
Microsoft.Net.Core.Component.SDK.2.1	.NET Core 2.1 development tools	15.8.27924.0	Optional
Microsoft.NetCore.ComponentGroup.DevelopmentTools	.NET Core 2.0 development tools	15.8.27729.1	Optional
Microsoft.NetCore.ComponentGroup.DevelopmentTools.2.1	.NET Core 2.1 development tools	15.8.27924.0	Optional
Microsoft.VisualStudio.Component.CodeClone	Code Clone	15.0.26208.0	Optional
Microsoft.VisualStudio.Component.CodeMap	Code Map	15.0.26208.0	Optional
Microsoft.VisualStudio.Component.Common.Azure.Tools	Connectivity and publishing tools	15.9.28107.0	Optional
Microsoft.VisualStudio.Component.DependencyValidation.Enterprise	Live Dependency Validation	15.0.26208.0	Optional
Microsoft.VisualStudio.Component.DockerTools	Container development tools	15.8.27906.1	Optional
Microsoft.VisualStudio.Component.DockerTools.BuildTools	Container development tools - Build Tools	15.7.27617.1	Optional
Microsoft.VisualStudio.Component.FSharp	F# language support	15.8.27825.0	Optional
Microsoft.VisualStudio.Component.FSharp.Desktop	F# desktop language support	15.8.27825.0	Optional
Microsoft.VisualStudio.Component.GraphDocument	DGML editor	15.0.27005.2	Optional
Microsoft.VisualStudio.Component.IISExpress	IIS Express	15.0.26208.0	Optional
Microsoft.VisualStudio.Component.JavaScript.Diagnostics	JavaScript diagnostics	15.8.27729.1	Optional

Component ID	Name	Version	Dependency Type
Microsoft.VisualStudio.Component.JavaScript.TypeScript	JavaScript and TypeScript language support	15.9.28125.51	Optional
Microsoft.VisualStudio.Component.NuGet	NuGet package manager	15.9.28016.0	Optional
Microsoft.VisualStudio.Component.SQL.ADAL	SQL ADAL runtime	15.6.27406.0	Optional
Microsoft.VisualStudio.Component.SQL.CMDUtils	SQL Server Command Line Utilities	15.0.26208.0	Optional
Microsoft.VisualStudio.Component.SQL.DataSources	Data sources for SQL Server support	15.0.26621.2	Optional
Microsoft.VisualStudio.Component.SQL.LocalDB.Runtime	SQL Server Express 2016 LocalDB	15.7.27617.1	Optional
Microsoft.VisualStudio.Component.SQL.NCLI	SQL Server Native Client	15.0.26208.0	Optional
Microsoft.VisualStudio.Component.SQL.SSDT	SQL Server Data Tools	15.9.28107.0	Optional
Microsoft.VisualStudio.Component.TypeScript.3.1	TypeScript 3.1 SDK	15.0.28218.60	Optional
Microsoft.VisualStudio.Component.Wcf.Tooling	Windows Communication Foundation	15.8.27924.0	Optional
Microsoft.VisualStudio.Component.Web	ASP.NET and web development tools	15.8.27825.0	Optional
Microsoft.VisualStudio.ComponentGroup.ArchitectureTools.Managed	Architecture and analysis tools	15.0.26208.0	Optional
Microsoft.VisualStudio.ComponentGroup.Web	ASP.NET and web development tools prerequisites	15.9.28219.51	Optional
Microsoft.VisualStudio.ComponentGroup.WebToolsExtensions	ASP.NET and web development	15.8.27825.0	Optional

Game development with Unity

ID: Microsoft.VisualStudio.Workload.ManagedGame

Description: Create 2D and 3D games with Unity, a powerful cross-platform development environment.

Components included by this workload

Component ID	Name	Version	Dependency Type
Microsoft.Net.Component.3.5.DeveloperTools	.NET Framework 3.5 development tools	15.6.27406.0	Required
Microsoft.Net.Component.4.7.1.TargetingPack	.NET Framework 4.7.1 targeting pack	15.6.27406.0	Required
Microsoft.VisualStudio.Component.NuGet	NuGet package manager	15.9.28016.0	Required
Microsoft.VisualStudio.Component.Roslyn.Compiler	C# and Visual Basic Roslyn compilers	15.6.27309.0	Required
Microsoft.VisualStudio.Component.Roslyn.LanguageServices	C# and Visual Basic	15.8.27729.1	Required
Microsoft.VisualStudio.Component.Static.Analysis.Tools	Static analysis tools	15.0.26208.0	Required
Microsoft.VisualStudio.Component.Unity	Visual Studio Tools for Unity	15.7.27617.1	Required
Component.UnityEngine.x64	Unity 2018.3 64-bit Editor	15.9.28307.271	Recommended
Component.UnityEngine.x86	Unity 5.6 32-bit Editor	15.6.27406.0	Recommended

Linux development with C++

ID: Microsoft.VisualStudio.Workload.NativeCrossPlat

Description: Create and debug applications running in a Linux environment.

Components included by this workload

Component ID	Name	Version	Dependency Type
Component.MDD.Linux	Visual C++ for Linux Development	15.6.27406.0	Required
Microsoft.VisualStudio.Component.VC.CoreLde	Visual Studio C++ core features	15.6.27406.0	Required
Microsoft.VisualStudio.Component.Windows10SDK	Windows Universal C Runtime	15.6.27406.0	Required
Component.Linux.CMake	Visual C++ tools for CMake and Linux	15.9.28307.102	Recommended
Microsoft.VisualStudio.Component.Static.Analysis.Tools	Static analysis tools	15.0.26208.0	Recommended
Microsoft.VisualStudio.Component.VC.Tools.x86.x64	VC++ 2017 version 15.9 v14.16 latest v141 tools	15.9.28230.55	Recommended

Component ID	Name	Version	Dependency Type
Microsoft.VisualStudio.Component.Windows10SDK.17763	Windows 10 SDK (10.0.17763.0)	15.9.28307.102	Recommended
Microsoft.VisualStudio.ComponentGroup.WebToolsExtensions	ASP.NET and web development	15.8.27825.0	Recommended
Component.MDD.Linux.GCC.arm	Embedded and IoT Development	15.6.27309.0	Optional

Desktop development with C++

ID: Microsoft.VisualStudio.Workload.NativeDesktop

Description: Build Windows desktop applications using the Microsoft C++ toolset, ATL, or MFC.

Components included by this workload

Component ID	Name	Version	Dependency Type
Microsoft.Component.MSBuild	MSBuild	15.7.27520.0	Required
Microsoft.VisualStudio.Component.ClassDesigner	Class Designer	15.0.26208.0	Required
Microsoft.VisualStudio.Component.CodeMap	Code Map	15.0.26208.0	Required
Microsoft.VisualStudio.Component.GraphDocument	DGML editor	15.0.27005.2	Required
Microsoft.VisualStudio.Component.Roslyn.Compiler	C# and Visual Basic Roslyn compilers	15.6.27309.0	Required
Microsoft.VisualStudio.Component.SQL.LocalDB.Runtime	SQL Server Express 2016 LocalDB	15.7.27617.1	Required
Microsoft.VisualStudio.Component.SQL.NCLI	SQL Server Native Client	15.0.26208.0	Required
Microsoft.VisualStudio.Component.TextTemplating	Text Template Transformation	15.0.26208.0	Required
Microsoft.VisualStudio.Component.VC.CoreEditor	Visual Studio C++ core features	15.6.27406.0	Required
Microsoft.VisualStudio.Component.VC.Redist.14.Latest	Visual C++ 2017 Redistributable Update	15.6.27406.0	Required
Microsoft.VisualStudio.ComponentGroup.ArchitectureTools.Native	Architecture tools for C++	15.0.26208.0	Required

Component ID	Name	Version	Dependency Type
Microsoft.VisualStudio.ComponentGroup.NativeDesktop.Core	Visual C++ core desktop features	15.8.27729.1	Required
Microsoft.VisualStudio.Component.Debugger.JustInTime	Just-In-Time debugger	15.0.27005.2	Recommended
Microsoft.VisualStudio.Component.Graphics.Tools	Graphics debugger and GPU profiler for DirectX	15.6.27406.0	Recommended
Microsoft.VisualStudio.Component.Graphics.Win81	Graphics Tools Windows 8.1 SDK	15.6.27406.0	Recommended
Microsoft.VisualStudio.Component.IntelliTrace.FrontEnd	IntelliTrace	15.8.27729.1	Recommended
Microsoft.VisualStudio.Component.NuGet	NuGet package manager	15.9.28016.0	Recommended
Microsoft.VisualStudio.Component.Static.Analysis.Tools	Static analysis tools	15.0.26208.0	Recommended
Microsoft.VisualStudio.Component.VC.ATL	Visual C++ ATL for x86 and x64	15.7.27625.0	Recommended
Microsoft.VisualStudio.Component.VC.CMake.Project	Visual C++ tools for CMake	15.9.28307.102	Recommended
Microsoft.VisualStudio.Component.VC.DiagnosticTools	C++ profiling tools	15.0.26823.1	Recommended
Microsoft.VisualStudio.Component.VC.TestAdapterForBoostTest	Test Adapter for Boost.Test	15.8.27906.1	Recommended
Microsoft.VisualStudio.Component.VC.TestAdapterForGoogleTest	Test Adapter for Google Test	15.8.27906.1	Recommended
Microsoft.VisualStudio.Component.VC.Tools.x86.x64	VC++ 2017 version 15.9 v14.16 latest v141 tools	15.9.28230.55	Recommended
Microsoft.VisualStudio.Component.Windows10SDK.17763	Windows 10 SDK (10.0.17763.0)	15.9.28307.102	Recommended
Microsoft.VisualStudio.ComponentGroup.WebToolsExtensions	ASP.NET and web development	15.8.27825.0	Recommended
Component.Incredibuild	Incredibuild - Build Acceleration	15.7.27617.1	Optional

Component ID	Name	Version	Dependency Type
Component.IncredibuildMenu	IncredibuildMenu	1.5.0.2	Optional
Microsoft.Component.VC.Runtime.UCRTSDK	Windows Universal CRT SDK	15.6.27309.0	Optional
Microsoft.Net.Component.4.6.1.SDK	.NET Framework 4.6.1 SDK	15.6.27406.0	Optional
Microsoft.Net.Component.4.6.1.TargetingPack	.NET Framework 4.6.1 targeting pack	15.6.27406.0	Optional
Microsoft.VisualStudio.Component.VC.140	VC++ 2015.3 v14.00 (v140) toolset for desktop	15.7.27617.1	Optional
Microsoft.VisualStudio.Component.VC.ATLMFC	Visual C++ MFC for x86 and x64	15.7.27625.0	Optional
Microsoft.VisualStudio.Component.VC.CLI.Support	C++/CLI support	15.6.27309.0	Optional
Microsoft.VisualStudio.Component.VC.Modules.x86.x64	Modules for Standard Library (experimental)	15.6.27309.0	Optional
Microsoft.VisualStudio.Component.Windows10SDK.10240	Windows 10 SDK (10.0.10240.0)	15.6.27406.0	Optional
Microsoft.VisualStudio.Component.Windows10SDK.10586	Windows 10 SDK (10.0.10586.0)	15.6.27406.0	Optional
Microsoft.VisualStudio.Component.Windows10SDK.14393	Windows 10 SDK (10.0.14393.0)	15.6.27406.0	Optional
Microsoft.VisualStudio.Component.Windows10SDK.15063/Desktop	Windows 10 SDK (10.0.15063.0) for Desktop C++ [x86 and x64]	15.6.27406.0	Optional
Microsoft.VisualStudio.Component.Windows10SDK.15063/UWP	Windows 10 SDK (10.0.15063.0) for UWP: C#, VB, JS	15.6.27406.0	Optional
Microsoft.VisualStudio.Component.Windows10SDK.15063/UWP.Native	Windows 10 SDK (10.0.15063.0) for UWP: C++	15.6.27406.0	Optional
Microsoft.VisualStudio.Component.Windows10SDK.16299/Desktop	Windows 10 SDK (10.0.16299.0) for Desktop C++ [x86 and x64]	15.6.27406.0	Optional
Microsoft.VisualStudio.Component.Windows10SDK.16299/Desktop.arm	Windows 10 SDK (10.0.16299.0) for Desktop C++ [ARM and ARM64]	15.6.27406.0	Optional

Component ID	Name	Version	Dependency Type
Microsoft.VisualStudio.Component.Windows10SDK.16299.UWP	Windows 10 SDK (10.0.16299.0) for UWP: C#, VB, JS	15.6.27406.0	Optional
Microsoft.VisualStudio.Component.Windows10SDK.16299.UWP.Native	Windows 10 SDK (10.0.16299.0) for UWP: C++	15.6.27406.0	Optional
Microsoft.VisualStudio.Component.Windows10SDK.17134	Windows 10 SDK (10.0.17134.0)	15.9.28307.102	Optional
Microsoft.VisualStudio.Component.Windows81SDK	Windows 8.1 SDK	15.6.27406.0	Optional
Microsoft.VisualStudio.Component.WinXP	Windows XP support for C++	15.8.27924.0	Optional
Microsoft.VisualStudio.ComponentGroup.NativeDesktop.Win81	Windows 8.1 SDK and UCRT SDK	15.6.27406.0	Optional
Microsoft.VisualStudio.ComponentGroup.NativeDesktop.WinXP	Windows XP support for C++	15.8.27705.0	Optional
Microsoft.VisualStudio.ComponentGroup.Windows10SDK.15063	Windows 10 SDK (10.0.15063.0)	15.8.27825.0	Optional
Microsoft.VisualStudio.ComponentGroup.Windows10SDK.16299	Windows 10 SDK (10.0.16299.0)	15.8.27825.0	Optional

Game development with C++

ID: Microsoft.VisualStudio.Workload.NativeGame

Description: Use the full power of C++ to build professional games powered by DirectX, Unreal, or Cocos2d.

Components included by this workload

Component ID	Name	Version	Dependency Type
Microsoft.VisualStudio.Component.Static.Analysis.Tools	Static analysis tools	15.0.26208.0	Required
Microsoft.VisualStudio.Component.VC.CoreIDE	Visual Studio C++ core features	15.6.27406.0	Required

Component ID	Name	Version	Dependency Type
Microsoft.VisualStudio.Component.VC.Redist.14.Latest	Visual C++ 2017 Redistributable Update	15.6.27406.0	Required
Microsoft.VisualStudio.Component.VC.Tools.x86.x64	VC++ 2017 version 15.9 v14.16 latest v141 tools	15.9.28230.55	Required
Microsoft.VisualStudio.Component.Windows10SDK	Windows Universal C Runtime	15.6.27406.0	Required
Microsoft.VisualStudio.Component.Graphics.Tools	Graphics debugger and GPU profiler for DirectX	15.6.27406.0	Recommended
Microsoft.VisualStudio.Component.Graphics.Win81	Graphics Tools Windows 8.1 SDK	15.6.27406.0	Recommended
Microsoft.VisualStudio.Component.IntelliTrace.FrontEnd	IntelliTrace	15.8.27729.1	Recommended
Microsoft.VisualStudio.Component.VC.DiagnosticTools	C++ profiling tools	15.0.26823.1	Recommended
Microsoft.VisualStudio.Component.Windows10SDK.17763	Windows 10 SDK (10.0.17763.0)	15.9.28307.102	Recommended
Component.Android.NDK.R12B	Android NDK (R12B)	12.1.10	Optional
Component.Android.SDK23.Private	Android SDK setup (API level 23) (local install for Mobile development with JavaScript / C++)	15.9.28016.0	Optional
Component.Ant	Apache Ant (1.9.3)	1.9.3.8	Optional
Component.Cocos	Cocos	15.0.26906.1	Optional
Component.Incredibuild	Incredibuild - Build Acceleration	15.7.27617.1	Optional
Component.IncredibuildMenu	IncredibuildMenu	1.5.0.2	Optional
Component.MDD.Android	C++ Android development tools	15.0.26606.0	Optional
Component.OpenJDK	Microsoft distribution OpenJDK	15.9.28125.51	Optional
Component.Unreal	Unreal Engine installer	15.8.27729.1	Optional
Component.Unreal.Android	Visual Studio Android support for Unreal Engine	15.9.28307.341	Optional

Component ID	Name	Version	Dependency Type
Microsoft.Component.VC.Runtime.UCRTSDK	Windows Universal CRT SDK	15.6.27309.0	Optional
Microsoft.Net.Component.4.5.1.TargetingPack	.NET Framework 4.5.1 targeting pack	15.6.27406.0	Optional
Microsoft.Net.Component.4.5.2.TargetingPack	.NET Framework 4.5.2 targeting pack	15.6.27406.0	Optional
Microsoft.Net.Component.4.5.TargetingPack	.NET Framework 4.5 targeting pack	15.6.27406.0	Optional
Microsoft.Net.Component.4.6.1.SDK	.NET Framework 4.6.1 SDK	15.6.27406.0	Optional
Microsoft.Net.Component.4.6.1.TargetingPack	.NET Framework 4.6.1 targeting pack	15.6.27406.0	Optional
Microsoft.Net.Component.4.6.TargetingPack	.NET Framework 4.6 targeting pack	15.6.27406.0	Optional
Microsoft.Net.Component.TargetingPack	.NET Framework 4 targeting pack	15.6.27406.0	Optional
Microsoft.Net.ComponentGroup.DevelopmentPrerequisites	.NET Framework 4.6.1 development tools	15.8.27825.0	Optional
Microsoft.Net.ComponentGroup.TargetingPacks.Common	.NET Framework 4 – 4.6 development tools	15.6.27406.0	Optional
Microsoft.VisualStudio.Component.NuGet.BuildTools	NuGet targets and build tasks	15.9.28016.0	Optional
Microsoft.VisualStudio.Component.Roslyn.Compiler	C# and Visual Basic Roslyn compilers	15.6.27309.0	Optional
Microsoft.VisualStudio.Component.Roslyn.LanguageServices	C# and Visual Basic	15.8.27729.1	Optional
Microsoft.VisualStudio.Component.Windows10SDK.10240	Windows 10 SDK (10.0.10240.0)	15.6.27406.0	Optional
Microsoft.VisualStudio.Component.Windows10SDK.10586	Windows 10 SDK (10.0.10586.0)	15.6.27406.0	Optional
Microsoft.VisualStudio.Component.Windows10SDK.14393	Windows 10 SDK (10.0.14393.0)	15.6.27406.0	Optional

Component ID	Name	Version	Dependency Type
Microsoft.VisualStudio.Component.Windows10SDK.15063.Desktop	Windows 10 SDK (10.0.15063.0) for Desktop C++ [x86 and x64]	15.6.27406.0	Optional
Microsoft.VisualStudio.Component.Windows10SDK.15063.UWP	Windows 10 SDK (10.0.15063.0) for UWP: C#, VB, JS	15.6.27406.0	Optional
Microsoft.VisualStudio.Component.Windows10SDK.15063.UWP.Native	Windows 10 SDK (10.0.15063.0) for UWP: C++	15.6.27406.0	Optional
Microsoft.VisualStudio.Component.Windows10SDK.16299.Desktop	Windows 10 SDK (10.0.16299.0) for Desktop C++ [x86 and x64]	15.6.27406.0	Optional
Microsoft.VisualStudio.Component.Windows10SDK.16299.Desktop.arm	Windows 10 SDK (10.0.16299.0) for Desktop C++ [ARM and ARM64]	15.6.27406.0	Optional
Microsoft.VisualStudio.Component.Windows10SDK.16299.UWP	Windows 10 SDK (10.0.16299.0) for UWP: C#, VB, JS	15.6.27406.0	Optional
Microsoft.VisualStudio.Component.Windows10SDK.16299.UWP.Native	Windows 10 SDK (10.0.16299.0) for UWP: C++	15.6.27406.0	Optional
Microsoft.VisualStudio.Component.Windows10SDK.17134	Windows 10 SDK (10.0.17134.0)	15.9.28307.102	Optional
Microsoft.VisualStudio.Component.Windows81SDK	Windows 8.1 SDK	15.6.27406.0	Optional
Microsoft.VisualStudio.ComponentGroup.NativeDesktop.Win81	Windows 8.1 SDK and UCRT SDK	15.6.27406.0	Optional
Microsoft.VisualStudio.ComponentGroup.Windows10SDK.15063	Windows 10 SDK (10.0.15063.0)	15.8.27825.0	Optional
Microsoft.VisualStudio.ComponentGroup.Windows10SDK.16299	Windows 10 SDK (10.0.16299.0)	15.8.27825.0	Optional

Mobile development with C++

ID: Microsoft.VisualStudio.Workload.NativeMobile

Description: Build cross-platform applications for iOS, Android or Windows using C++.

Components included by this workload

Component ID	Name	Version	Dependency Type
Component.Android.SDK19.Private	Android SDK setup (API level 19) (local install for Mobile development with JavaScript / C++)	15.9.28107.0	Required
Component.Android.SDK21.Private	Android SDK setup (API level 21) (local install for Mobile development with JavaScript / C++)	15.9.28016.0	Required
Component.Android.SDK22.Private	Android SDK setup (API level 22) (local install for Mobile development with JavaScript / C++)	15.9.28016.0	Required
Component.Android.SDK23.Private	Android SDK setup (API level 23) (local install for Mobile development with JavaScript / C++)	15.9.28016.0	Required
Component.Android.SDK25.Private	Android SDK setup (API level 25) (local install for Mobile development with JavaScript / C++)	15.9.28016.0	Required
Component.OpenJDK	Microsoft distribution OpenJDK	15.9.28125.51	Required
Microsoft.VisualStudio.Component.VC.Corede	Visual Studio C++ core features	15.6.27406.0	Required
Component.Android.NDK.R15C	Android NDK (R15C)	15.2.1	Recommended
Component.Ant	Apache Ant (1.9.3)	1.9.3.8	Recommended
Component.MDD.Android	C++ Android development tools	15.0.26606.0	Recommended
Component.Android.NDK.R12B	Android NDK (R12B)	12.1.10	Optional
Component.Android.NDK.R12B_3264	Android NDK (R12B) (32bit)	12.1.11	Optional
Component.Android.NDK.R13B	Android NDK (R13B)	13.1.7	Optional
Component.Android.NDK.R13B_3264	Android NDK (R13B) (32bit)	13.1.8	Optional
Component.Android.NDK.R15C_3264	Android NDK (R15C) (32bit)	15.2.1	Optional

COMPONENT ID	NAME	VERSION	DEPENDENCY TYPE
Component.Google.Android.Emulator.API23.Private	Google Android Emulator (API Level 23) (local install)	15.6.27413.0	Optional
Component.HAXM.Private	Intel Hardware Accelerated Execution Manager (HAXM) (local install)	15.9.28307.421	Optional
Component.Incredibuild	Incredibuild - Build Acceleration	15.7.27617.1	Optional
Component.IncredibuildMenu	IncredibuildMenu	1.5.0.2	Optional
Component.MDD.iOS	C++ iOS development tools	15.0.26621.2	Optional

.NET Core cross-platform development

ID: Microsoft.VisualStudio.Workload.NetCoreTools

Description: Build cross-platform applications using .NET Core, ASP.NET Core, HTML/JavaScript, and Containers including Docker support.

Components included by this workload

COMPONENT ID	NAME	VERSION	DEPENDENCY TYPE
Component.Microsoft.VisualStudio.RazorExtension	Razor Language Services	15.0.26720.2	Required
Component.Microsoft.Web.LibraryManager	Library Manager	15.8.27705.0	Required
Component.WebSocket	WebSocket4Net	15.0.26606.0	Required
Microsoft.Component.ClickOnce	ClickOnce Publishing	15.8.27825.0	Required
Microsoft.Component_MSBUILD	MSBuild	15.7.27520.0	Required
Microsoft.Net.Component.4.5.2.TargetingPack	.NET Framework 4.5.2 targeting pack	15.6.27406.0	Required
Microsoft.Net.Component.4.5.TargetingPack	.NET Framework 4.5 targeting pack	15.6.27406.0	Required
Microsoft.Net.Component.4.6.1.SDK	.NET Framework 4.6.1 SDK	15.6.27406.0	Required
Microsoft.Net.Component.4.6.1.TargetingPack	.NET Framework 4.6.1 targeting pack	15.6.27406.0	Required

Component ID	Name	Version	Dependency Type
Microsoft.Net.ComponentGroup.DevelopmentPrerequisites	.NET Framework 4.6.1 development tools	15.8.27825.0	Required
Microsoft.Net.Core.Component.SDK.2.1	.NET Core 2.1 development tools	15.8.27924.0	Required
Microsoft.NetCore.ComponentGroup.DevelopmentTools.2.1	.NET Core 2.1 development tools	15.8.27924.0	Required
Microsoft.NetCore.ComponentGroup.Web.2.1	.NET Core 2.1 development tools	15.8.27924.0	Required
Microsoft.VisualStudio.Component.Common.Azure.Tools	Connectivity and publishing tools	15.9.28107.0	Required
Microsoft.VisualStudio.Component.DockerTools	Container development tools	15.8.27906.1	Required
Microsoft.VisualStudio.Component.DockerTools.BuildTools	Container development tools - Build Tools	15.7.27617.1	Required
Microsoft.VisualStudio.Component.FSharp	F# language support	15.8.27825.0	Required
Microsoft.VisualStudio.Component.FSharp.WebTemplates	F# language support for web projects	15.9.28307.421	Required
Microsoft.VisualStudio.Component.IIExpress	IIS Express	15.0.26208.0	Required
Microsoft.VisualStudio.Component.JavaScript.Diagnostics	JavaScript diagnostics	15.8.27729.1	Required
Microsoft.VisualStudio.Component.JavaScript.TypeScript	JavaScript and TypeScript language support	15.9.28125.51	Required
Microsoft.VisualStudio.Component.ManagedDesktop.Core	Managed Desktop Workload Core	15.8.27729.1	Required
Microsoft.VisualStudio.Component.NuGet	NuGet package manager	15.9.28016.0	Required
Microsoft.VisualStudio.Component.PortableLibrary	.NET Portable Library targeting pack	15.6.27309.0	Required
Microsoft.VisualStudio.Component.Roslyn.Compiler	C# and Visual Basic Roslyn compilers	15.6.27309.0	Required

Component ID	Name	Version	Dependency Type
Microsoft.VisualStudio.Component.Roslyn.LanguageServices	C# and Visual Basic	15.8.27729.1	Required
Microsoft.VisualStudio.Component.SQL.ADAL	SQL ADAL runtime	15.6.27406.0	Required
Microsoft.VisualStudio.Component.SQL.CLR	CLR data types for SQL Server	15.0.26208.0	Required
Microsoft.VisualStudio.Component.SQL.CMDUtils	SQL Server Command Line Utilities	15.0.26208.0	Required
Microsoft.VisualStudio.Component.SQL.DataSources	Data sources for SQL Server support	15.0.26621.2	Required
Microsoft.VisualStudio.Component.SQL.LocalDB.Runtime	SQL Server Express 2016 LocalDB	15.7.27617.1	Required
Microsoft.VisualStudio.Component.SQL.NCLI	SQL Server Native Client	15.0.26208.0	Required
Microsoft.VisualStudio.Component.SQL.SSDT	SQL Server Data Tools	15.9.28107.0	Required
Microsoft.VisualStudio.Component.Static.Analysis.Tools	Static analysis tools	15.0.26208.0	Required
Microsoft.VisualStudio.Component.TextTemplating	Text Template Transformation	15.0.26208.0	Required
Microsoft.VisualStudio.Component.TypeScript.3.1	TypeScript 3.1 SDK	15.0.28218.60	Required
Microsoft.VisualStudio.Component.VisualStudioData	Data sources and service references	15.6.27406.0	Required
Microsoft.VisualStudio.ComponentGroup.Web	ASP.NET and web development tools prerequisites	15.9.28219.51	Required
Microsoft.VisualStudio.ComponentGroup.WebToolsExtensions	ASP.NET and web development	15.8.27825.0	Required
Component.Microsoft.VisualStudio.Web.AzureFunctions	Microsoft Azure WebJobs Tools	15.7.27617.1	Recommended
Microsoft.VisualStudio.Component.AppInsights.Tools	Developer Analytics tools	15.8.27825.0	Recommended
Microsoft.VisualStudio.Component.Azure.AuthoringTools	Azure Authoring Tools	15.9.28307.421	Recommended

Component ID	Name	Version	Dependency Type
Microsoft.VisualStudio.Component.Azure.ClientLibs	Azure libraries for .NET	15.0.26208.0	Recommended
Microsoft.VisualStudio.Component.Azure.Compute.Emulator	Azure Compute Emulator	15.9.28307.421	Recommended
Microsoft.VisualStudio.Component.Azure.Storage.Emulator	Azure Storage Emulator	15.9.28125.51	Recommended
Microsoft.VisualStudio.Component.CloudExplorer	Cloud Explorer	15.9.28230.55	Recommended
Microsoft.VisualStudio.Component.Debugger.Snapshot	Snapshot Debugger	15.8.28010.0	Recommended
Microsoft.VisualStudio.Component.DiagnosticTools	.NET profiling tools	15.8.27729.1	Recommended
Microsoft.VisualStudio.Component.IntelliTrace.FrontEnd	IntelliTrace	15.8.27729.1	Recommended
Microsoft.VisualStudio.Component.LiveUnitTesting	Live Unit Testing	15.0.26720.2	Recommended
Microsoft.VisualStudio.Component.Web	ASP.NET and web development tools	15.8.27825.0	Recommended
Microsoft.VisualStudio.Component.WebDeploy	Web Deploy	15.8.27729.1	Recommended
Microsoft.VisualStudio.ComponentGroup.AzureFunctions	Microsoft Azure WebJobs Tools	15.7.27617.1	Recommended
Microsoft.VisualStudio.ComponentGroup.Web.CloudTools	Cloud tools for web development	15.8.27729.1	Recommended
Microsoft.Net.Core.Component.SDK	.NET Core 2.0 development tools	15.6.27406.0	Optional
Microsoft.Net.Core.Component.SDK.1x	.NET Core 1.0 - 1.1 development tools	15.6.27406.0	Optional
Microsoft.NetCore.1x.ComponentGroup.Web	.NET Core 1.0 - 1.1 development tools for Web	15.6.27406.0	Optional
Microsoft.NetCore.ComponentGroup.DevelopmentTools	.NET Core 2.0 development tools	15.8.27729.1	Optional
Microsoft.NetCore.ComponentGroup.Web	.NET Core 2.0 development tools	15.7.27625.0	Optional

Component ID	Name	Version	Dependency Type
Microsoft.VisualStudio.ComponentGroup.IISDevelopment	Development time IIS support	15.9.28219.51	Optional

Mobile development with .NET

ID: Microsoft.VisualStudio.Workload.NetCrossPlat

Description: Build cross-platform applications for iOS, Android or Windows using Xamarin.

Components included by this workload

Component ID	Name	Version	Dependency Type
Component.Xamarin	Xamarin	15.8.27906.1	Required
Component.Xamarin.RemotedSimulator	Xamarin Remoted Simulator	15.6.27323.2	Required
Microsoft.Component.MSBuild	MSBuild	15.7.27520.0	Required
Microsoft.Net.Component.4.6.1.SDK	.NET Framework 4.6.1 SDK	15.6.27406.0	Required
Microsoft.Net.Component.4.6.1.TargetingPack	.NET Framework 4.6.1 targeting pack	15.6.27406.0	Required
Microsoft.Net.ComponentGroup.DevelopmentPrerequisites	.NET Framework 4.6.1 development tools	15.8.27825.0	Required
Microsoft.Net.Core.Component.SDK	.NET Core 2.0 development tools	15.6.27406.0	Required
Microsoft.NetCore.ComponentGroup.DevelopmentTools	.NET Core 2.0 development tools	15.8.27729.1	Required
Microsoft.VisualStudio.Component.FSharp	F# language support	15.8.27825.0	Required
Microsoft.VisualStudio.Component.Merq	Common Xamarin internal tools	15.8.27924.0	Required
Microsoft.VisualStudio.Component.MonoDebugger	Mono debugger	15.0.26720.2	Required
Microsoft.VisualStudio.Component.NuGet	NuGet package manager	15.9.28016.0	Required
Microsoft.VisualStudio.Component.PortableLibrary	.NET Portable Library targeting pack	15.6.27309.0	Required
Microsoft.VisualStudio.Component.Roslyn.Compiler	C# and Visual Basic Roslyn compilers	15.6.27309.0	Required

Component ID	Name	Version	Dependency Type
Microsoft.VisualStudio.Component.Roslyn.LanguageServices	C# and Visual Basic	15.8.27729.1	Required
Microsoft.VisualStudio.Component.Static.Analysis.Tools	Static analysis tools	15.0.26208.0	Required
Microsoft.VisualStudio.ComponentGroup.WebToolsExtensions.TemplateEngine	ASP.NET templating engine	15.8.27729.1	Required
Component.Android.SDK27	Android SDK setup (API level 27)	15.9.28016.0	Recommended
Component.Google.Android.Emulator.API27	Google Android Emulator (API Level 27)	15.9.28307.421	Recommended
Component.HAXM	Intel Hardware Accelerated Execution Manager (HAXM) (global install)	15.9.28307.421	Recommended
Component.OpenJDK	Microsoft distribution OpenJDK	15.9.28125.51	Recommended
Component.Xamarin.Profiler	Xamarin Profiler	15.0.27005.2	Recommended
Component.Xamarin.Inspector	Xamarin Workbooks	15.0.26606.0	Optional
Microsoft.Component.ClickOnce	ClickOnce Publishing	15.8.27825.0	Optional
Microsoft.Component.NetFX.Native	.NET Native	15.0.26208.0	Optional
Microsoft.VisualStudio.Component.AppInsights.Tools	Developer Analytics tools	15.8.27825.0	Optional
Microsoft.VisualStudio.Component.CodeClone	Code Clone	15.0.26208.0	Optional
Microsoft.VisualStudio.Component.CodeMap	Code Map	15.0.26208.0	Optional
Microsoft.VisualStudio.Component.DependencyValidation.Enterprise	Live Dependency Validation	15.0.26208.0	Optional
Microsoft.VisualStudio.Component.DiagnosticTools	.NET profiling tools	15.8.27729.1	Optional
Microsoft.VisualStudio.Component.GraphDocument	DGML editor	15.0.27005.2	Optional

COMPONENT ID	NAME	VERSION	DEPENDENCY TYPE
Microsoft.VisualStudio.Component.Graphics	Image and 3D model editors	15.6.27406.0	Optional
Microsoft.VisualStudio.Component.SQL.CLR	CLR data types for SQL Server	15.0.26208.0	Optional
Microsoft.VisualStudio.Component.SQL.LocalDB.Runtime	SQL Server Express 2016 LocalDB	15.7.27617.1	Optional
Microsoft.VisualStudio.Component.SQL.NCLI	SQL Server Native Client	15.0.26208.0	Optional
Microsoft.VisualStudio.Component.VisualStudioData	Data sources and service references	15.6.27406.0	Optional
Microsoft.VisualStudio.Component.Windows10SDK.17763	Windows 10 SDK (10.0.17763.0)	15.9.28307.102	Optional
Microsoft.VisualStudio.ComponentGroup.ArchitectureTools.Managed	Architecture and analysis tools	15.0.26208.0	Optional
Microsoft.VisualStudio.ComponentGroup.UWP.Xamarin	Universal Windows Platform tools for Xamarin	15.9.28307.102	Optional

ASP.NET and web development

ID: Microsoft.VisualStudio.Workload.NetWeb

Description: Build web applications using ASP.NET, ASP.NET Core, HTML/JavaScript, and Containers including Docker support.

Components included by this workload

COMPONENT ID	NAME	VERSION	DEPENDENCY TYPE
Component.Microsoft.VisualStudio.RazorExtension	Razor Language Services	15.0.26720.2	Required
Component.Microsoft.Web.LibraryManager	Library Manager	15.8.27705.0	Required
Component.WebSocket	WebSocket4Net	15.0.26606.0	Required
Microsoft.Component.ClickOnce	ClickOnce Publishing	15.8.27825.0	Required
Microsoft.Component_MSBuild	MSBuild	15.7.27520.0	Required
Microsoft.Net.Component.4.5.2.TargetingPack	.NET Framework 4.5.2 targeting pack	15.6.27406.0	Required

Component ID	Name	Version	Dependency Type
Microsoft.Net.Component.4.5.TargetingPack	.NET Framework 4.5 targeting pack	15.6.27406.0	Required
Microsoft.Net.Component.4.6.1.SDK	.NET Framework 4.6.1 SDK	15.6.27406.0	Required
Microsoft.Net.Component.4.6.1.TargetingPack	.NET Framework 4.6.1 targeting pack	15.6.27406.0	Required
Microsoft.Net.ComponentGroup.DevelopmentPrerequisites	.NET Framework 4.6.1 development tools	15.8.27825.0	Required
Microsoft.Net.Core.Component.SDK.2.1	.NET Core 2.1 development tools	15.8.27924.0	Required
Microsoft.NetCore.ComponentGroup.DevelopmentTools.2.1	.NET Core 2.1 development tools	15.8.27924.0	Required
Microsoft.NetCore.ComponentGroup.Web.2.1	.NET Core 2.1 development tools	15.8.27924.0	Required
Microsoft.VisualStudio.Component.Common.Azure.Tools	Connectivity and publishing tools	15.9.28107.0	Required
Microsoft.VisualStudio.Component.DockerTools	Container development tools	15.8.27906.1	Required
Microsoft.VisualStudio.Component.DockerTools.BuildTools	Container development tools - Build Tools	15.7.27617.1	Required
Microsoft.VisualStudio.Component.FSharp	F# language support	15.8.27825.0	Required
Microsoft.VisualStudio.Component.FSharp.WebTemplates	F# language support for web projects	15.9.28307.421	Required
Microsoft.VisualStudio.Component.IIExpress	IIS Express	15.0.26208.0	Required
Microsoft.VisualStudio.Component.JavaScript.Diagnostics	JavaScript diagnostics	15.8.27729.1	Required
Microsoft.VisualStudio.Component.JavaScript.TypeScript	JavaScript and TypeScript language support	15.9.28125.51	Required
Microsoft.VisualStudio.Component.ManagedDesktop.Core	Managed Desktop Workload Core	15.8.27729.1	Required
Microsoft.VisualStudio.Component.NuGet	NuGet package manager	15.9.28016.0	Required

Component ID	Name	Version	Dependency Type
Microsoft.VisualStudio.Component.PortableLibrary	.NET Portable Library targeting pack	15.6.27309.0	Required
Microsoft.VisualStudio.Component.Roslyn.Compiler	C# and Visual Basic Roslyn compilers	15.6.27309.0	Required
Microsoft.VisualStudio.Component.Roslyn.LanguageServices	C# and Visual Basic	15.8.27729.1	Required
Microsoft.VisualStudio.Component.SQL.ADAL	SQL ADAL runtime	15.6.27406.0	Required
Microsoft.VisualStudio.Component.SQL.CLR	CLR data types for SQL Server	15.0.26208.0	Required
Microsoft.VisualStudio.Component.SQL.CMDUtils	SQL Server Command Line Utilities	15.0.26208.0	Required
Microsoft.VisualStudio.Component.SQL.DataSources	Data sources for SQL Server support	15.0.26621.2	Required
Microsoft.VisualStudio.Component.SQL.LocalDB.Runtime	SQL Server Express 2016 LocalDB	15.7.27617.1	Required
Microsoft.VisualStudio.Component.SQL.NCLI	SQL Server Native Client	15.0.26208.0	Required
Microsoft.VisualStudio.Component.SQL.SSDT	SQL Server Data Tools	15.9.28107.0	Required
Microsoft.VisualStudio.Component.Static.Analysis.Tools	Static analysis tools	15.0.26208.0	Required
Microsoft.VisualStudio.Component.TextTemplating	Text Template Transformation	15.0.26208.0	Required
Microsoft.VisualStudio.Component.TypeScript.3.1	TypeScript 3.1 SDK	15.0.28218.60	Required
Microsoft.VisualStudio.Component.VisualStudioData	Data sources and service references	15.6.27406.0	Required
Microsoft.VisualStudio.Component.Web	ASP.NET and web development tools	15.8.27825.0	Required
Microsoft.VisualStudio.ComponentGroup.Web	ASP.NET and web development tools prerequisites	15.9.28219.51	Required
Microsoft.VisualStudio.ComponentGroup.WebToolsExtensions	ASP.NET and web development	15.8.27825.0	Required

Component ID	Name	Version	Dependency Type
Component.Microsoft.VisualStudio.Web.AzureFunctions	Microsoft Azure WebJobs Tools	15.7.27617.1	Recommended
Microsoft.Net.Component.4.5.1.TargetingPack	.NET Framework 4.5.1 targeting pack	15.6.27406.0	Recommended
Microsoft.Net.Component.4.6.TargetingPack	.NET Framework 4.6 targeting pack	15.6.27406.0	Recommended
Microsoft.Net.Component.4.TargetingPack	.NET Framework 4 targeting pack	15.6.27406.0	Recommended
Microsoft.Net.ComponentGroup.TargetingPacks.Common	.NET Framework 4 – 4.6 development tools	15.6.27406.0	Recommended
Microsoft.VisualStudio.Component.AppInsights.Tools	Developer Analytics tools	15.8.27825.0	Recommended
Microsoft.VisualStudio.Component.AspNet45	Advanced ASP.NET features	15.7.27625.0	Recommended
Microsoft.VisualStudio.Component.Azure.AuthoringTools	Azure Authoring Tools	15.9.28307.421	Recommended
Microsoft.VisualStudio.Component.Azure.ClientLibs	Azure libraries for .NET	15.0.26208.0	Recommended
Microsoft.VisualStudio.Component.Azure.Compute Emulator	Azure Compute Emulator	15.9.28307.421	Recommended
Microsoft.VisualStudio.Component.Azure.Storage Emulator	Azure Storage Emulator	15.9.28125.51	Recommended
Microsoft.VisualStudio.Component.CloudExplorer	Cloud Explorer	15.9.28230.55	Recommended
Microsoft.VisualStudio.Component.Debugger.Snapshot	Snapshot Debugger	15.8.28010.0	Recommended
Microsoft.VisualStudio.Component.DiagnosticTools	.NET profiling tools	15.8.27729.1	Recommended
Microsoft.VisualStudio.Component.EntityFramework	Entity Framework 6 tools	15.6.27406.0	Recommended
Microsoft.VisualStudio.Component.IntelliTrace.FrontEnd	IntelliTrace	15.8.27729.1	Recommended
Microsoft.VisualStudio.Component.LiveUnitTesting	Live Unit Testing	15.0.26720.2	Recommended

Component ID	Name	Version	Dependency Type
Microsoft.VisualStudio.Component.Wcf.Tooling	Windows Communication Foundation	15.8.27924.0	Recommended
Microsoft.VisualStudio.Component.WebDeploy	Web Deploy	15.8.27729.1	Recommended
Microsoft.VisualStudio.ComponentGroup.AzureFunctions	Microsoft Azure WebJobs Tools	15.7.27617.1	Recommended
Microsoft.VisualStudio.ComponentGroup.Web.CloudTools	Cloud tools for web development	15.8.27729.1	Recommended
Microsoft.Net.Component.4.6.2.SDK	.NET Framework 4.6.2 SDK	15.6.27406.0	Optional
Microsoft.Net.Component.4.6.2.TargetingPack	.NET Framework 4.6.2 targeting pack	15.6.27406.0	Optional
Microsoft.Net.Component.4.7.1.SDK	.NET Framework 4.7.1 SDK	15.6.27406.0	Optional
Microsoft.Net.Component.4.7.1.TargetingPack	.NET Framework 4.7.1 targeting pack	15.6.27406.0	Optional
Microsoft.Net.Component.4.7.2.SDK	.NET Framework 4.7.2 SDK	15.8.27825.0	Optional
Microsoft.Net.Component.4.7.2.TargetingPack	.NET Framework 4.7.2 targeting pack	15.8.27825.0	Optional
Microsoft.Net.Component.4.7.SDK	.NET Framework 4.7 SDK	15.6.27406.0	Optional
Microsoft.Net.Component.4.7.TargetingPack	.NET Framework 4.7 targeting pack	15.6.27406.0	Optional
Microsoft.Net.ComponentGroup.4.6.2.DeveloperTools	.NET Framework 4.6.2 development tools	15.6.27406.0	Optional
Microsoft.Net.ComponentGroup.4.7.1.DeveloperTools	.NET Framework 4.7.1 development tools	15.6.27406.0	Optional
Microsoft.Net.ComponentGroup.4.7.2.DeveloperTools	.NET Framework 4.7.2 development tools	15.8.27825.0	Optional
Microsoft.Net.ComponentGroup.4.7.DeveloperTools	.NET Framework 4.7 development tools	15.6.27406.0	Optional
Microsoft.Net.Core.Component.SDK	.NET Core 2.0 development tools	15.6.27406.0	Optional
Microsoft.Net.Core.Component.SDK.1x	.NET Core 1.0 - 1.1 development tools	15.6.27406.0	Optional

Component ID	Name	Version	Dependency Type
Microsoft.NetCore.1x.ComponentGroup.Web	.NET Core 1.0 - 1.1 development tools for Web	15.6.27406.0	Optional
Microsoft.NetCore.ComponentGroup.DevelopmentTools	.NET Core 2.0 development tools	15.8.27729.1	Optional
Microsoft.NetCore.ComponentGroup.Web	.NET Core 2.0 development tools	15.7.27625.0	Optional
Microsoft.VisualStudio.Component.CodeClone	Code Clone	15.0.26208.0	Optional
Microsoft.VisualStudio.Component.CodeMap	Code Map	15.0.26208.0	Optional
Microsoft.VisualStudio.Component.DependencyValidation.Enterprise	Live Dependency Validation	15.0.26208.0	Optional
Microsoft.VisualStudio.Component.GraphDocument	DGML editor	15.0.27005.2	Optional
Microsoft.VisualStudio.Component.TestTools.WebLoadTest	Web performance and load testing tools	15.8.27729.1	Optional
Microsoft.VisualStudio.ComponentGroup.ArchitectureTools.Managed	Architecture and analysis tools	15.0.26208.0	Optional
Microsoft.VisualStudio.ComponentGroup.IISDevelopment	Development time IIS support	15.9.28219.51	Optional
Microsoft.VisualStudio.Web.Mvc4.ComponentGroup	ASP.NET MVC 4	15.6.27406.0	Optional

Node.js development

ID: Microsoft.VisualStudio.Workload.Node

Description: Build scalable network applications using Node.js, an asynchronous event-driven JavaScript runtime.

Components included by this workload

Component ID	Name	Version	Dependency Type
Component.WebSocket	WebSocket4Net	15.0.26606.0	Required
Microsoft.VisualStudio.Component.JavaScript.Diagnostics	JavaScript diagnostics	15.8.27729.1	Required
Microsoft.VisualStudio.Component.JavaScript.TypeScript	JavaScript and TypeScript language support	15.9.28125.51	Required

Component ID	Name	Version	Dependency Type
Microsoft.VisualStudio.Component.Node.Build	Node.js MSBuild support	15.8.27825.0	Required
Microsoft.VisualStudio.Component.Node.Tools	Node.js development support	15.8.27825.0	Required
Microsoft.VisualStudio.Component.NuGet	NuGet package manager	15.9.28016.0	Required
Microsoft.VisualStudio.Component.TestTools.Core	Testing tools core features	15.7.27520.0	Required
Microsoft.VisualStudio.Component.TypeScript.3.1	TypeScript 3.1 SDK	15.0.28218.60	Required
Microsoft.VisualStudio.ComponentGroup.WebToolsExtensions	ASP.NET and web development	15.8.27825.0	Required
Microsoft.VisualStudio.Component.WebDeploy	Web Deploy	15.8.27729.1	Recommended
Microsoft.VisualStudio.Component.AppInsights.Tools	Developer Analytics tools	15.8.27825.0	Optional
Microsoft.VisualStudio.Component.Common.Azure.Tools	Connectivity and publishing tools	15.9.28107.0	Optional
Microsoft.VisualStudio.Component.Static.Analysis.Tools	Static analysis tools	15.0.26208.0	Optional
Microsoft.VisualStudio.Component.VC.CoreRedist	Visual Studio C++ core features	15.6.27406.0	Optional
Microsoft.VisualStudio.Component.VC.Tools.x86.x64	VC++ 2017 version 15.9 v14.16 latest v141 tools	15.9.28230.55	Optional

Office/SharePoint development

ID: Microsoft.VisualStudio.Workload.Office

Description: Create Office and SharePoint add-ins, SharePoint solutions, and VSTO add-ins using C#, VB, and JavaScript.

Components included by this workload

Component ID	Name	Version	Dependency Type
Component.Microsoft.VisualStudio.RazorExtension	Razor Language Services	15.0.26720.2	Required
Component.Microsoft.Web.LibraryManager	Library Manager	15.8.27705.0	Required

Component ID	Name	Version	Dependency Type
Component.WebSocket	WebSocket4Net	15.0.26606.0	Required
Microsoft.Component.ClickOnce	ClickOnce Publishing	15.8.27825.0	Required
Microsoft.Component_MSBUILD	MSBuild	15.7.27520.0	Required
Microsoft.Net.Component.4.5.2.TargetingPack	.NET Framework 4.5.2 targeting pack	15.6.27406.0	Required
Microsoft.Net.Component.4.5.TargetingPack	.NET Framework 4.5 targeting pack	15.6.27406.0	Required
Microsoft.Net.Component.4.6.1.SDK	.NET Framework 4.6.1 SDK	15.6.27406.0	Required
Microsoft.Net.Component.4.6.1.TargetingPack	.NET Framework 4.6.1 targeting pack	15.6.27406.0	Required
Microsoft.Net.Component.4.TargetingPack	.NET Framework 4 targeting pack	15.6.27406.0	Required
Microsoft.Net.ComponentGroup.DevelopmentPrerequisites	.NET Framework 4.6.1 development tools	15.8.27825.0	Required
Microsoft.Net.Core.Component.SDK.2.1	.NET Core 2.1 development tools	15.8.27924.0	Required
Microsoft.VisualStudio.Component.AppInsights.Tools	Developer Analytics tools	15.8.27825.0	Required
Microsoft.VisualStudio.Component.Common.Azure.Tools	Connectivity and publishing tools	15.9.28107.0	Required
Microsoft.VisualStudio.Component.DockerTools	Container development tools	15.8.27906.1	Required
Microsoft.VisualStudio.Component.DockerTools.BuildTools	Container development tools - Build Tools	15.7.27617.1	Required
Microsoft.VisualStudio.Component.IISExpress	IIS Express	15.0.26208.0	Required
Microsoft.VisualStudio.Component.JavaScript.Diagnostics	JavaScript diagnostics	15.8.27729.1	Required
Microsoft.VisualStudio.Component.JavaScript.TypeScript	JavaScript and TypeScript language support	15.9.28125.51	Required

Component ID	Name	Version	Dependency Type
Microsoft.VisualStudio.Component.ManagedDesktop.Core	Managed Desktop Workload Core	15.8.27729.1	Required
Microsoft.VisualStudio.Component.ManagedDesktop.Prerequisites	.NET desktop development tools	15.7.27625.0	Required
Microsoft.VisualStudio.Component.NuGet	NuGet package manager	15.9.28016.0	Required
Microsoft.VisualStudio.Component.PortableLibrary	.NET Portable Library targeting pack	15.6.27309.0	Required
Microsoft.VisualStudio.Component.Roslyn.Compiler	C# and Visual Basic Roslyn compilers	15.6.27309.0	Required
Microsoft.VisualStudio.Component.Roslyn.LanguageServices	C# and Visual Basic	15.8.27729.1	Required
Microsoft.VisualStudio.Component.Sharepoint.Tools	Office Developer Tools for Visual Studio	15.8.27924.0	Required
Microsoft.VisualStudio.Component.SQL.ADAL	SQL ADAL runtime	15.6.27406.0	Required
Microsoft.VisualStudio.Component.SQL.CLR	CLR data types for SQL Server	15.0.26208.0	Required
Microsoft.VisualStudio.Component.SQL.CMDUtils	SQL Server Command Line Utilities	15.0.26208.0	Required
Microsoft.VisualStudio.Component.SQL.DataSources	Data sources for SQL Server support	15.0.26621.2	Required
Microsoft.VisualStudio.Component.SQL.LocalDB.Runtime	SQL Server Express 2016 LocalDB	15.7.27617.1	Required
Microsoft.VisualStudio.Component.SQL.NCLI	SQL Server Native Client	15.0.26208.0	Required
Microsoft.VisualStudio.Component.SQL.SSDT	SQL Server Data Tools	15.9.28107.0	Required
Microsoft.VisualStudio.Component.Static.Analysis.Tools	Static analysis tools	15.0.26208.0	Required
Microsoft.VisualStudio.Component.TextTemplating	Text Template Transformation	15.0.26208.0	Required
Microsoft.VisualStudio.Component.TypeScript.3.1	TypeScript 3.1 SDK	15.0.28218.60	Required

Component ID	Name	Version	Dependency Type
Microsoft.VisualStudio.Component.VisualStudioData	Data sources and service references	15.6.27406.0	Required
Microsoft.VisualStudio.Component.Wcf.Tooling	Windows Communication Foundation	15.8.27924.0	Required
Microsoft.VisualStudio.Component.Web	ASP.NET and web development tools	15.8.27825.0	Required
Microsoft.VisualStudio.Component.Workflow	Windows Workflow Foundation	15.8.27825.0	Required
Microsoft.VisualStudio.ComponentGroup.Web	ASP.NET and web development tools prerequisites	15.9.28219.51	Required
Microsoft.VisualStudio.ComponentGroup.WebToolsExtensions	ASP.NET and web development	15.8.27825.0	Required
Microsoft.VisualStudio.Component.TeamOffice	Visual Studio Tools for Office (VSTO)	15.7.27625.0	Recommended
Microsoft.VisualStudio.Component.WebDeploy	Web Deploy	15.8.27729.1	Recommended
Microsoft.Net.Component.4.6.2.SDK	.NET Framework 4.6.2 SDK	15.6.27406.0	Optional
Microsoft.Net.Component.4.6.2.TargetingPack	.NET Framework 4.6.2 targeting pack	15.6.27406.0	Optional
Microsoft.Net.Component.4.7.1.SDK	.NET Framework 4.7.1 SDK	15.6.27406.0	Optional
Microsoft.Net.Component.4.7.1.TargetingPack	.NET Framework 4.7.1 targeting pack	15.6.27406.0	Optional
Microsoft.Net.Component.4.7.2.SDK	.NET Framework 4.7.2 SDK	15.8.27825.0	Optional
Microsoft.Net.Component.4.7.2.TargetingPack	.NET Framework 4.7.2 targeting pack	15.8.27825.0	Optional
Microsoft.Net.Component.4.7.SDK	.NET Framework 4.7 SDK	15.6.27406.0	Optional
Microsoft.Net.Component.4.7.TargetingPack	.NET Framework 4.7 targeting pack	15.6.27406.0	Optional
Microsoft.Net.ComponentGroup.4.6.2.DeveloperTools	.NET Framework 4.6.2 development tools	15.6.27406.0	Optional

Component ID	Name	Version	Dependency Type
Microsoft.Net.ComponentGroup.4.7.1.DeveloperTools	.NET Framework 4.7.1 development tools	15.6.27406.0	Optional
Microsoft.Net.ComponentGroup.4.7.2.DeveloperTools	.NET Framework 4.7.2 development tools	15.8.27825.0	Optional
Microsoft.Net.ComponentGroup.4.7.DeveloperTools	.NET Framework 4.7 development tools	15.6.27406.0	Optional
Microsoft.VisualStudio.Component.IntelliTrace.FrontEnd	IntelliTrace	15.8.27729.1	Optional

Python development

ID: Microsoft.VisualStudio.Workload.Python

Description: Editing, debugging, interactive development and source control for Python.

Components included by this workload

Component ID	Name	Version	Dependency Type
Microsoft.Component.PythonTools	Python language support	15.0.26823.1	Required
Component.CPython3.x64	Python 3 64-bit (3.6.6)	3.6.6	Recommended
Microsoft.Component.CookiecutterTools	Cookiecutter template support	15.0.26621.2	Recommended
Microsoft.Component.PythonTools.Web	Python web support	15.9.28107.0	Recommended
Microsoft.VisualStudio.Component.Common.Azure.Tools	Connectivity and publishing tools	15.9.28107.0	Recommended
Microsoft.VisualStudio.Component.JavaScript.TypeScript	JavaScript and TypeScript language support	15.9.28125.51	Recommended
Microsoft.VisualStudio.Component.SQL.CLR	CLR data types for SQL Server	15.0.26208.0	Recommended
Microsoft.VisualStudio.Component.TypeScript.3.1	TypeScript 3.1 SDK	15.0.28218.60	Recommended
Microsoft.VisualStudio.Component.VisualStudioData	Data sources and service references	15.6.27406.0	Recommended
Microsoft.VisualStudio.Component.WebDeploy	Web Deploy	15.8.27729.1	Recommended
Microsoft.VisualStudio.ComponentGroup.WebToolsExtensions	ASP.NET and web development	15.8.27825.0	Recommended

Component ID	Name	Version	Dependency Type
Component.Anaconda2.x64	Anaconda2 64-bit (5.2.0)	5.2.0	Optional
Component.Anaconda2.x86	Anaconda2 32-bit (5.2.0)	5.2.0	Optional
Component.Anaconda3.x64	Anaconda3 64-bit (5.2.0)	5.2.0	Optional
Component.Anaconda3.x86	Anaconda3 32-bit (5.2.0)	5.2.0	Optional
Component.CPython2.x64	Python 2 64-bit (2.7.14)	2.7.14	Optional
Component.CPython2.x86	Python 2 32-bit (2.7.14)	2.7.14	Optional
Component.CPython3.x86	Python 3 32-bit (3.6.6)	3.6.6	Optional
Component.Microsoft.VisualStudio.RazorExtension	Razor Language Services	15.0.26720.2	Optional
Component.Microsoft.Web.LibraryManager	Library Manager	15.8.27705.0	Optional
Component.WebSocket	WebSocket4Net	15.0.26606.0	Optional
Microsoft.Component.ClickOnce	ClickOnce Publishing	15.8.27825.0	Optional
Microsoft.Component.MSBuild	MSBuild	15.7.27520.0	Optional
Microsoft.Component.NetFX.Native	.NET Native	15.0.26208.0	Optional
Microsoft.Component.PythonTools.UWP	Python IoT support	15.0.26606.0	Optional
Microsoft.Component.VC.Runtime.UCRTSDK	Windows Universal CRT SDK	15.6.27309.0	Optional
Microsoft.ComponentGroup.PythonTools.NativeDevelopment	Python native development tools	15.9.28307.102	Optional
Microsoft.Net.Component.4.5.2.TargetingPack	.NET Framework 4.5.2 targeting pack	15.6.27406.0	Optional
Microsoft.Net.Component.4.5.TargetingPack	.NET Framework 4.5 targeting pack	15.6.27406.0	Optional
Microsoft.Net.Component.4.6.1.SDK	.NET Framework 4.6.1 SDK	15.6.27406.0	Optional
Microsoft.Net.Component.4.6.1.TargetingPack	.NET Framework 4.6.1 targeting pack	15.6.27406.0	Optional

Component ID	Name	Version	Dependency Type
Microsoft.Net.ComponentGroup.DevelopmentPrerequisites	.NET Framework 4.6.1 development tools	15.8.27825.0	Optional
Microsoft.Net.Core.Component.SDK.2.1	.NET Core 2.1 development tools	15.8.27924.0	Optional
Microsoft.VisualStudio.Component.AppInsights.Tools	Developer Analytics tools	15.8.27825.0	Optional
Microsoft.VisualStudio.Component.Azure.AuthoringTools	Azure Authoring Tools	15.9.28307.421	Optional
Microsoft.VisualStudio.Component.Azure.ClientLibs	Azure libraries for .NET	15.0.26208.0	Optional
Microsoft.VisualStudio.Component.Azure.Compute.Emulator	Azure Compute Emulator	15.9.28307.421	Optional
Microsoft.VisualStudio.Component.Azure.Storage.Emulator	Azure Storage Emulator	15.9.28125.51	Optional
Microsoft.VisualStudio.Component.Azure.Waverton	Azure Cloud Services core tools	15.9.28107.0	Optional
Microsoft.VisualStudio.Component.Azure.Waverton.BuildTools	Azure Cloud Services build tools	15.7.27617.1	Optional
Microsoft.VisualStudio.Component.ClassDesigner	Class Designer	15.0.26208.0	Optional
Microsoft.VisualStudio.Component.CodeClone	Code Clone	15.0.26208.0	Optional
Microsoft.VisualStudio.Component.CodeMap	Code Map	15.0.26208.0	Optional
Microsoft.VisualStudio.Component.DiagnosticTools	.NET profiling tools	15.8.27729.1	Optional
Microsoft.VisualStudio.Component.DockerTools	Container development tools	15.8.27906.1	Optional
Microsoft.VisualStudio.Component.DockerTools.BuildTools	Container development tools - Build Tools	15.7.27617.1	Optional
Microsoft.VisualStudio.Component.GraphDocument	DGML editor	15.0.27005.2	Optional

Component ID	Name	Version	Dependency Type
Microsoft.VisualStudio.Component.Graphics	Image and 3D model editors	15.6.27406.0	Optional
Microsoft.VisualStudio.Component.Graphics.Tools	Graphics debugger and GPU profiler for DirectX	15.6.27406.0	Optional
Microsoft.VisualStudio.Component.Graphics.Win81	Graphics Tools Windows 8.1 SDK	15.6.27406.0	Optional
Microsoft.VisualStudio.Component.IISExpress	IIS Express	15.0.26208.0	Optional
Microsoft.VisualStudio.Component.IntelliTrace.FrontEnd	IntelliTrace	15.8.27729.1	Optional
Microsoft.VisualStudio.Component.JavaScript.Diagnostics	JavaScript diagnostics	15.8.27729.1	Optional
Microsoft.VisualStudio.Component.ManagedDesktop.Core	Managed Desktop Workload Core	15.8.27729.1	Optional
Microsoft.VisualStudio.Component.NuGet	NuGet package manager	15.9.28016.0	Optional
Microsoft.VisualStudio.Component.PortableLibrary	.NET Portable Library targeting pack	15.6.27309.0	Optional
Microsoft.VisualStudio.Component.Roslyn.Compiler	C# and Visual Basic Roslyn compilers	15.6.27309.0	Optional
Microsoft.VisualStudio.Component.Roslyn.LanguageServices	C# and Visual Basic	15.8.27729.1	Optional
Microsoft.VisualStudio.Component.SQL.ADAL	SQL ADAL runtime	15.6.27406.0	Optional
Microsoft.VisualStudio.Component.SQL.CMDUtils	SQL Server Command Line Utilities	15.0.26208.0	Optional
Microsoft.VisualStudio.Component.SQL.DataSources	Data sources for SQL Server support	15.0.26621.2	Optional
Microsoft.VisualStudio.Component.SQL.LocalDB.Runtime	SQL Server Express 2016 LocalDB	15.7.27617.1	Optional
Microsoft.VisualStudio.Component.SQL.NCLI	SQL Server Native Client	15.0.26208.0	Optional
Microsoft.VisualStudio.Component.SQL.SSDT	SQL Server Data Tools	15.9.28107.0	Optional

Component ID	Name	Version	Dependency Type
Microsoft.VisualStudio.Component.Static.Analysis.Tools	Static analysis tools	15.0.26208.0	Optional
Microsoft.VisualStudio.Component.TextTemplating	Text Template Transformation	15.0.26208.0	Optional
Microsoft.VisualStudio.Component.VC.140	VC++ 2015.3 v14.00 (v140) toolset for desktop	15.7.27617.1	Optional
Microsoft.VisualStudio.Component.VC.Corelde	Visual Studio C++ core features	15.6.27406.0	Optional
Microsoft.VisualStudio.Component.VC.DiagnosticTools	C++ profiling tools	15.0.26823.1	Optional
Microsoft.VisualStudio.Component.VC.Tools.x86.x64	VC++ 2017 version 15.9 v14.16 latest v141 tools	15.9.28230.55	Optional
Microsoft.VisualStudio.Component.Web	ASP.NET and web development tools	15.8.27825.0	Optional
Microsoft.VisualStudio.Component.Windows10SDK	Windows Universal C Runtime	15.6.27406.0	Optional
Microsoft.VisualStudio.Component.Windows10SDK.1058.6	Windows 10 SDK (10.0.10586.0)	15.6.27406.0	Optional
Microsoft.VisualStudio.Component.Windows10SDK.1776.3	Windows 10 SDK (10.0.17763.0)	15.9.28307.102	Optional
Microsoft.VisualStudio.Component.Windows81SDK	Windows 8.1 SDK	15.6.27406.0	Optional
Microsoft.VisualStudio.ComponentGroup.Web	ASP.NET and web development tools prerequisites	15.9.28219.51	Optional

Universal Windows Platform development

ID: Microsoft.VisualStudio.Workload.Universal

Description: Create applications for the Universal Windows Platform with C#, VB, JavaScript, or optionally C++.

Components included by this workload

Component ID	Name	Version	Dependency Type
Component.WebSocket	WebSocket4Net	15.0.26606.0	Required
Microsoft.Component.ClickOnce	ClickOnce Publishing	15.8.27825.0	Required

Component ID	Name	Version	Dependency Type
Microsoft.Component.NetFX.Native	.NET Native	15.0.26208.0	Required
Microsoft.ComponentGroup.Blend	Blend for Visual Studio	15.6.27406.0	Required
Microsoft.Net.Component.4.5.TargetingPack	.NET Framework 4.5 targeting pack	15.6.27406.0	Required
Microsoft.Net.Component.4.6.1.SDK	.NET Framework 4.6.1 SDK	15.6.27406.0	Required
Microsoft.Net.Core.Component.SDK.2.1	.NET Core 2.1 development tools	15.8.27924.0	Required
Microsoft.VisualStudio.Component.AppInsights.Tools	Developer Analytics tools	15.8.27825.0	Required
Microsoft.VisualStudio.Component.DiagnosticTools	.NET profiling tools	15.8.27729.1	Required
Microsoft.VisualStudio.Component.Graphics	Image and 3D model editors	15.6.27406.0	Required
Microsoft.VisualStudio.Component.JavaScript.Diagnostics	JavaScript diagnostics	15.8.27729.1	Required
Microsoft.VisualStudio.Component.JavaScript.TypeScript	JavaScript and TypeScript language support	15.9.28125.51	Required
Microsoft.VisualStudio.Component.NuGet	NuGet package manager	15.9.28016.0	Required
Microsoft.VisualStudio.Component.PortableLibrary	.NET Portable Library targeting pack	15.6.27309.0	Required
Microsoft.VisualStudio.Component.Roslyn.Compiler	C# and Visual Basic Roslyn compilers	15.6.27309.0	Required
Microsoft.VisualStudio.Component.Roslyn.LanguageServices	C# and Visual Basic	15.8.27729.1	Required
Microsoft.VisualStudio.Component.SQL.CLR	CLR data types for SQL Server	15.0.26208.0	Required
Microsoft.VisualStudio.Component.Static.Analysis.Tools	Static analysis tools	15.0.26208.0	Required
Microsoft.VisualStudio.Component.TypeScript.3.1	TypeScript 3.1 SDK	15.0.28218.60	Required
Microsoft.VisualStudio.Component.UWP.Support	Universal Windows Platform tools	15.9.28119.51	Required

Component ID	Name	Version	Dependency Type
Microsoft.VisualStudio.Component.VisualStudioData	Data sources and service references	15.6.27406.0	Required
Microsoft.VisualStudio.Component.Windows10SDK.17763	Windows 10 SDK (10.0.17763.0)	15.9.28307.102	Required
Microsoft.VisualStudio.ComponentGroup.UWP.Cordova	Universal Windows Platform tools for Cordova	15.9.28307.102	Required
Microsoft.VisualStudio.ComponentGroup.UWP.NetCoreAndStandard	.NET Native and .NET Standard	15.8.27906.1	Required
Microsoft.VisualStudio.ComponentGroup.UWP.Xamarin	Universal Windows Platform tools for Xamarin	15.9.28307.102	Required
Microsoft.VisualStudio.ComponentGroup.WebToolsExtensions	ASP.NET and web development	15.8.27825.0	Required
Microsoft.VisualStudio.Component.IntelliTrace.FrontEnd	IntelliTrace	15.8.27729.1	Recommended
Microsoft.Component.VC.Runtime.OSSupport	Visual C++ runtime for UWP	15.6.27406.0	Optional
Microsoft.Net.Component.4.7.2.SDK	.NET Framework 4.7.2 SDK	15.8.27825.0	Optional
Microsoft.VisualStudio.Component.CodeClone	Code Clone	15.0.26208.0	Optional
Microsoft.VisualStudio.Component.CodeMap	Code Map	15.0.26208.0	Optional
Microsoft.VisualStudio.Component.DependencyValidation.Enterprise	Live Dependency Validation	15.0.26208.0	Optional
Microsoft.VisualStudio.Component.GraphDocument	DGML editor	15.0.27005.2	Optional
Microsoft.VisualStudio.Component.Graphics.Tools	Graphics debugger and GPU profiler for DirectX	15.6.27406.0	Optional
Microsoft.VisualStudio.Component.Graphics.Win81	Graphics Tools Windows 8.1 SDK	15.6.27406.0	Optional
Microsoft.VisualStudio.Component.Phone.Emulator.15254	Windows 10 Mobile Emulator (Fall Creators Update)	15.0.27406.0	Optional

Component ID	Name	Version	Dependency Type
Microsoft.VisualStudio.Component.SQL.LocalDB.Runtime	SQL Server Express 2016 LocalDB	15.7.27617.1	Optional
Microsoft.VisualStudio.Component.SQL.NCLI	SQL Server Native Client	15.0.26208.0	Optional
Microsoft.VisualStudio.Component.UWP.VC.ARM64	C++ Universal Windows Platform tools for ARM64	15.0.28125.51	Optional
Microsoft.VisualStudio.Component.VC.CoreLDE	Visual Studio C++ core features	15.6.27406.0	Optional
Microsoft.VisualStudio.Component.VC.Tools.ARM	Visual C++ compilers and libraries for ARM	15.8.27825.0	Optional
Microsoft.VisualStudio.Component.VC.Tools.ARM64	Visual C++ compilers and libraries for ARM64	15.9.28230.55	Optional
Microsoft.VisualStudio.Component.VC.Tools.x86.x64	VC++ 2017 version 15.9 v14.16 latest v141 tools	15.9.28230.55	Optional
Microsoft.VisualStudio.Component.Windows10SDK.10240	Windows 10 SDK (10.0.10240.0)	15.6.27406.0	Optional
Microsoft.VisualStudio.Component.Windows10SDK.10586	Windows 10 SDK (10.0.10586.0)	15.6.27406.0	Optional
Microsoft.VisualStudio.Component.Windows10SDK.14393	Windows 10 SDK (10.0.14393.0)	15.6.27406.0	Optional
Microsoft.VisualStudio.Component.Windows10SDK.15063/Desktop	Windows 10 SDK (10.0.15063.0) for Desktop C++ [x86 and x64]	15.6.27406.0	Optional
Microsoft.VisualStudio.Component.Windows10SDK.15063/UWP	Windows 10 SDK (10.0.15063.0) for UWP: C#, VB, JS	15.6.27406.0	Optional
Microsoft.VisualStudio.Component.Windows10SDK.15063/UWP.Native	Windows 10 SDK (10.0.15063.0) for UWP: C++	15.6.27406.0	Optional
Microsoft.VisualStudio.Component.Windows10SDK.16299/Desktop	Windows 10 SDK (10.0.16299.0) for Desktop C++ [x86 and x64]	15.6.27406.0	Optional
Microsoft.VisualStudio.Component.Windows10SDK.16299/Desktop.arm	Windows 10 SDK (10.0.16299.0) for Desktop C++ [ARM and ARM64]	15.6.27406.0	Optional

Component ID	Name	Version	Dependency Type
Microsoft.VisualStudio.Component.Windows10SDK.16299.UWP	Windows 10 SDK (10.0.16299.0) for UWP: C#, VB, JS	15.6.27406.0	Optional
Microsoft.VisualStudio.Component.Windows10SDK.16299.UWP.Native	Windows 10 SDK (10.0.16299.0) for UWP: C++	15.6.27406.0	Optional
Microsoft.VisualStudio.Component.Windows10SDK.17134	Windows 10 SDK (10.0.17134.0)	15.9.28307.102	Optional
Microsoft.VisualStudio.Component.Windows10SDK.IoPOverUsb	USB Device Connectivity	15.9.28307.102	Optional
Microsoft.VisualStudio.ComponentGroup.ArchitectureTools.Managed	Architecture and analysis tools	15.0.26208.0	Optional
Microsoft.VisualStudio.ComponentGroup.UWP.VC	C++ Universal Windows Platform tools	15.9.28307.102	Optional
Microsoft.VisualStudio.ComponentGroup.Windows10SDK.15063	Windows 10 SDK (10.0.15063.0)	15.8.27825.0	Optional
Microsoft.VisualStudio.ComponentGroup.Windows10SDK.16299	Windows 10 SDK (10.0.16299.0)	15.8.27825.0	Optional

Visual Studio extension development

ID: Microsoft.VisualStudio.Workload.VisualStudioExtension

Description: Create add-ons and extensions for Visual Studio, including new commands, code analyzers and tool windows.

Components included by this workload

Component ID	Name	Version	Dependency Type
Microsoft.Component.ClickOnce	ClickOnce Publishing	15.8.27825.0	Required
Microsoft.Component_MSBUILD	MSBuild	15.7.27520.0	Required
Microsoft.Net.Component.4.6.1.SDK	.NET Framework 4.6.1 SDK	15.6.27406.0	Required
Microsoft.Net.Component.4.6.1.TargetingPack	.NET Framework 4.6.1 targeting pack	15.6.27406.0	Required

Component ID	Name	Version	Dependency Type
Microsoft.Net.Component.4.6.TargetingPack	.NET Framework 4.6 targeting pack	15.6.27406.0	Required
Microsoft.Net.ComponentGroup.DevelopmentPrerequisites	.NET Framework 4.6.1 development tools	15.8.27825.0	Required
Microsoft.VisualStudio.Component.NuGet	NuGet package manager	15.9.28016.0	Required
Microsoft.VisualStudio.Component.PortableLibrary	.NET Portable Library targeting pack	15.6.27309.0	Required
Microsoft.VisualStudio.Component.Roslyn.Compiler	C# and Visual Basic Roslyn compilers	15.6.27309.0	Required
Microsoft.VisualStudio.Component.Roslyn.LanguageServices	C# and Visual Basic	15.8.27729.1	Required
Microsoft.VisualStudio.Component.Static.Analysis.Tools	Static analysis tools	15.0.26208.0	Required
Microsoft.VisualStudio.Component.VSSDK	Visual Studio SDK	15.8.27729.1	Required
Microsoft.VisualStudio.ComponentGroup.VisualStudioExtension.Prerequisites	Visual Studio extension development prerequisites	15.7.27625.0	Required
Microsoft.VisualStudio.Component.DiagnosticTools	.NET profiling tools	15.8.27729.1	Recommended
Microsoft.VisualStudio.Component.IntelliTrace.FrontEnd	IntelliTrace	15.8.27729.1	Recommended
Microsoft.VisualStudio.Component.TextTemplating	Text Template Transformation	15.0.26208.0	Recommended
Component.Dotfuscator	PreEmptive Protection - Dotfuscator	15.0.26208.0	Optional
Microsoft.Component.CodeAnalysis.SDK	.NET Compiler Platform SDK	15.0.27729.1	Optional
Microsoft.Component.VC.Runtime.OSSupport	Visual C++ runtime for UWP	15.6.27406.0	Optional
Microsoft.VisualStudio.Component.AppInsights.Tools	Developer Analytics tools	15.8.27825.0	Optional
Microsoft.VisualStudio.Component.ClassDesigner	Class Designer	15.0.26208.0	Optional

Component ID	Name	Version	Dependency Type
Microsoft.VisualStudio.Component.CodeClone	Code Clone	15.0.26208.0	Optional
Microsoft.VisualStudio.Component.CodeMap	Code Map	15.0.26208.0	Optional
Microsoft.VisualStudio.Component.DependencyValidation.Enterprise	Live Dependency Validation	15.0.26208.0	Optional
Microsoft.VisualStudio.Component.DslTools	Modeling SDK	15.0.27005.2	Optional
Microsoft.VisualStudio.Component.GraphDocument	DGML editor	15.0.27005.2	Optional
Microsoft.VisualStudio.Component.SQL.LocalDB.Runtime	SQL Server Express 2016 LocalDB	15.7.27617.1	Optional
Microsoft.VisualStudio.Component.SQL.NCLI	SQL Server Native Client	15.0.26208.0	Optional
Microsoft.VisualStudio.Component.VC.ATL	Visual C++ ATL for x86 and x64	15.7.27625.0	Optional
Microsoft.VisualStudio.Component.VC.ATLMFC	Visual C++ MFC for x86 and x64	15.7.27625.0	Optional
Microsoft.VisualStudio.Component.VC.Corelde	Visual Studio C++ core features	15.6.27406.0	Optional
Microsoft.VisualStudio.Component.VC.Tools.x86.x64	VC++ 2017 version 15.9 v14.16 latest v141 tools	15.9.28230.55	Optional
Microsoft.VisualStudio.ComponentGroup.ArchitectureTools.Managed	Architecture and analysis tools	15.0.26208.0	Optional

Mobile development with JavaScript

ID: Microsoft.VisualStudio.Workload.WebCrossPlat

Description: Build Android, iOS and UWP apps using Tools for Apache Cordova.

Components included by this workload

Component ID	Name	Version	Dependency Type
Component.CordovaToolset.6.3.1	Cordova 6.3.1 toolset	15.7.27625.0	Required
Component.WebSocket	WebSocket4Net	15.0.26606.0	Required

Component ID	Name	Version	Dependency Type
Microsoft.VisualStudio.Component.Cordova	Mobile development with JavaScript core features	15.0.26606.0	Required
Microsoft.VisualStudio.Component.JavaScript.Diagnostics	JavaScript diagnostics	15.8.27729.1	Required
Microsoft.VisualStudio.Component.JavaScript.ProjectSystem	JavaScript ProjectSystem and Shared Tooling	15.0.26606.0	Required
Microsoft.VisualStudio.Component.JavaScript.TypeScript	JavaScript and TypeScript language support	15.9.28125.51	Required
Microsoft.VisualStudio.Component.TypeScript.2.3	TypeScript 2.3 SDK	15.8.27729.1	Required
Microsoft.VisualStudio.Component.TypeScript.3.1	TypeScript 3.1 SDK	15.0.28218.60	Required
Microsoft.VisualStudio.ComponentGroup.WebToolsExtensions	ASP.NET and web development	15.8.27825.0	Required
Component.Android.SDK23.Private	Android SDK setup (API level 23) (local install for Mobile development with JavaScript / C++)	15.9.28016.0	Optional
Component.Google.Android.Emulator.API23.Private	Google Android Emulator (API Level 23) (local install)	15.6.27413.0	Optional
Component.HAXM.Private	Intel Hardware Accelerated Execution Manager (HAXM) (local install)	15.9.28307.421	Optional
Component.OpenJDK	Microsoft distribution OpenJDK	15.9.28125.51	Optional
Microsoft.Component.ClickOnce	ClickOnce Publishing	15.8.27825.0	Optional
Microsoft.Component.NetFX.Native	.NET Native	15.0.26208.0	Optional
Microsoft.VisualStudio.Component.AppInsights.Tools	Developer Analytics tools	15.8.27825.0	Optional
Microsoft.VisualStudio.Component.DiagnosticTools	.NET profiling tools	15.8.27729.1	Optional
Microsoft.VisualStudio.Component.Git	Git for Windows	15.0.26208.0	Optional

Component ID	Name	Version	Dependency Type
Microsoft.VisualStudio.Component.Graphics	Image and 3D model editors	15.6.27406.0	Optional
Microsoft.VisualStudio.Component.Phone Emulator.15254	Windows 10 Mobile Emulator (Fall Creators Update)	15.0.27406.0	Optional
Microsoft.VisualStudio.Component.SQL.CLR	CLR data types for SQL Server	15.0.26208.0	Optional
Microsoft.VisualStudio.Component.VisualStudioData	Data sources and service references	15.6.27406.0	Optional
Microsoft.VisualStudio.Component.Windows10SDK.17763	Windows 10 SDK (10.0.17763.0)	15.9.28307.102	Optional
Microsoft.VisualStudio.ComponentGroup.UWP.Cordova	Universal Windows Platform tools for Cordova	15.9.28307.102	Optional

Unaffiliated components

These are components that are not included with any workload, but may be selected as an individual component.

Component ID	Name	Version
Component.Android.Emulator	Visual Studio Emulator for Android	15.6.27413.0
Component.Android.NDK.R11C	Android NDK (R11C)	11.3.14
Component.Android.NDK.R11C_3264	Android NDK (R11C) (32bit)	11.3.16
Component.Android.SDK23	Android SDK setup (API level 23) (global install)	15.9.28107.0
Component.Android.SDK25	Android SDK setup (API level 25)	15.9.28107.0
Component.GitHub.VisualStudio	GitHub extension for Visual Studio	2.5.2.2500
Component.Google.Android.Emulator.API23.V2	Google Android Emulator (API Level 23) (global install)	15.6.27413.0
Component.Google.Android.Emulator.API25	Google Android Emulator (API Level 25)	15.7.27604.0
Microsoft.Component.Blend.SDK.WPF	Blend for Visual Studio SDK for .NET	15.6.27406.0
Microsoft.Component.HelpViewer	Help Viewer	15.9.28307.421
Microsoft.VisualStudio.Component.LinqToSql	LINQ to SQL tools	15.6.27406.0

Component ID	Name	Version
Microsoft.VisualStudio.Component.Phone.Emulator	Windows 10 Mobile Emulator (Anniversary Edition)	15.6.27406.0
Microsoft.VisualStudio.Component.Phone.Emulator.15063	Windows 10 Mobile Emulator (Creators Update)	15.6.27406.0
Microsoft.VisualStudio.Component.Runtime.Node.x86.6.4.0	Runtime for components based on Node.js v6.4.0 (x86)	15.7.27617.1
Microsoft.VisualStudio.Component.Runtime.Node.x86.7.4.0	Runtime for components based on Node.js v7.4.0 (x86)	15.7.27617.1
Microsoft.VisualStudio.Component.TestTools.CodedUITest	Coded UI test	15.0.26606.0
Microsoft.VisualStudio.Component.TestTools.FeedbackClient	Microsoft Feedback Client	15.6.27406.0
Microsoft.VisualStudio.Component.TestTools.MicrosoftTestManager	Microsoft Test Manager	15.6.27406.0
Microsoft.VisualStudio.Component.TypeScript.2.0	TypeScript 2.0 SDK	15.8.27729.1
Microsoft.VisualStudio.Component.TypeScript.2.1	TypeScript 2.1 SDK	15.8.27729.1
Microsoft.VisualStudio.Component.TypeScript.2.2	TypeScript 2.2 SDK	15.8.27729.1
Microsoft.VisualStudio.Component.TypeScript.2.5	TypeScript 2.5 SDK	15.6.27406.0
Microsoft.VisualStudio.Component.TypeScript.2.6	TypeScript 2.6 SDK	15.0.27729.1
Microsoft.VisualStudio.Component.TypeScript.2.7	TypeScript 2.7 SDK	15.0.27729.1
Microsoft.VisualStudio.Component.TypeScript.2.8	TypeScript 2.8 SDK	15.0.27729.1
Microsoft.VisualStudio.Component.TypeScript.2.9	TypeScript 2.9 SDK	15.0.27924.0
Microsoft.VisualStudio.Component.TypeScript.3.0	TypeScript 3.0 SDK	15.0.27924.0
Microsoft.VisualStudio.Component.VC.ATL.ARM	Visual C++ ATL for ARM	15.7.27625.0
Microsoft.VisualStudio.Component.VC.ATL.ARM.Spectre	Visual C++ ATL for ARM with Spectre Mitigations	15.7.27625.0

Component ID	Name	Version
Microsoft.VisualStudio.Component.VC.ATL.ARM64	Visual C++ ATL for ARM64	15.7.27625.0
Microsoft.VisualStudio.Component.VC.ATL.ARM64.Spectre	Visual C++ ATL for ARM64 with Spectre Mitigations	15.7.27625.0
Microsoft.VisualStudio.Component.VC.ATL.Spectre	Visual C++ ATL (x86/x64) with Spectre Mitigations	15.7.27625.0
Microsoft.VisualStudio.Component.VC.ATLMFC.Spectre	Visual C++ MFC for x86/x64 with Spectre Mitigations	15.7.27625.0
Microsoft.VisualStudio.Component.VC.ClangC2	Clang/C2 (experimental)	15.7.27520.0
Microsoft.VisualStudio.Component.VC.MFC.ARM	Visual C++ MFC for ARM	15.7.27625.0
Microsoft.VisualStudio.Component.VC.MFC.ARM.Spectre	Visual C++ MFC for ARM with Spectre Mitigations	15.7.27625.0
Microsoft.VisualStudio.Component.VC.MFC.ARM64	Visual C++ MFC for ARM64	15.7.27625.0
Microsoft.VisualStudio.Component.VC.MFC.ARM64.Spectre	Visual C++ MFC support for ARM64 with Spectre Mitigations	15.7.27625.0
Microsoft.VisualStudio.Component.VC.Runtimes.ARM.Spectre	VC++ 2017 version 15.9 v14.16 Libs for Spectre (ARM)	15.9.28230.55
Microsoft.VisualStudio.Component.VC.Runtimes.ARM64.Spectre	VC++ 2017 version 15.9 v14.16 Libs for Spectre (ARM64)	15.9.28230.55
Microsoft.VisualStudio.Component.VC.Runtimes.x86.x64.Spectre	VC++ 2017 version 15.9 v14.16 Libs for Spectre (x86 and x64)	15.9.28230.55
Microsoft.VisualStudio.Component.VC.Tools.14.11	VC++ 2017 version 15.4 v14.11 toolset	15.0.27924.0
Microsoft.VisualStudio.Component.VC.Tools.14.12	VC++ 2017 version 15.5 v14.12 toolset	15.0.27924.0
Microsoft.VisualStudio.Component.VC.Tools.14.13	VC++ 2017 version 15.6 v14.13 toolset	15.0.27924.0
Microsoft.VisualStudio.Component.VC.Tools.14.14	VC++ 2017 version 15.7 v14.14 toolset	15.0.27924.0
Microsoft.VisualStudio.Component.VC.Tools.14.15	VC++ 2017 version 15.8 v14.15 toolset	15.0.28230.55

When you set dependencies in your VSIX manifest, you must specify Component IDs only. Use the tables on this page to determine the minimum component dependencies. In some scenarios, this might mean that you specify only one component from a workload. In other scenarios, it might mean that you specify multiple components.

from a single workload or multiple components from multiple workloads. For more information, see the [How to: Migrate extensibility projects to Visual Studio](#) page.

For more information about how to use these IDs, see the [Use command-line parameters to install Visual Studio](#) page. And, for a list of workload and component IDs for other products, see the [Visual Studio workload and component IDs](#) page.

Visual Studio core editor (included with Visual Studio Enterprise 2019)

ID: Microsoft.VisualStudio.Workload.CoreEditor

Description: The Visual Studio core shell experience, including syntax-aware code editing, source code control and work item management.

Components included by this workload

COMPONENT ID	NAME	VERSION	DEPENDENCY TYPE
Microsoft.VisualStudio.Component.CoreEditor	Visual Studio core editor	16.1.28811.260	Required
Microsoft.VisualStudio.Component.StartPageExperiment.Cpp	Visual Studio Start Page for C++ Users	16.0.28315.86	Optional

Azure development

ID: Microsoft.VisualStudio.Workload.Azure

Description: Azure SDKs, tools, and projects for developing cloud apps and creating resources using .NET Core and .NET Framework. Also includes tools for containerizing your application, including Docker support.

Components included by this workload

COMPONENT ID	NAME	VERSION	DEPENDENCY TYPE
Component.Microsoft.VisualStudio.RazorExtension	Razor Language Services	16.0.28714.129	Required
Component.Microsoft.VisualStudio.Web.AzureFunctions	Azure WebJobs Tools	16.0.28714.129	Required
Component.Microsoft.Web.LibraryManager	Library Manager	16.0.28315.86	Required
Microsoft.Component.MSBuild	MSBuild	16.4.29429.68	Required
Microsoft.Net.Component.4.5.2.TargetingPack	.NET Framework 4.5.2 targeting pack	16.0.28517.75	Required
Microsoft.Net.Component.4.5.TargetingPack	.NET Framework 4.5 targeting pack	16.0.28517.75	Required
Microsoft.Net.Component.4.6.1.TargetingPack	.NET Framework 4.6.1 targeting pack	16.0.28517.75	Required

Component ID	Name	Version	Dependency Type
Microsoft.Net.Component.4.7.2.TargetingPack	.NET Framework 4.7.2 targeting pack	16.0.28517.75	Required
Microsoft.Net.Component.4.8.SDK	.NET Framework 4.8 SDK	16.4.29313.120	Required
Microsoft.Net.ComponentGroup.DevelopmentPrerequisites	.NET Framework 4.7.2 development tools	16.3.29207.166	Required
Microsoft.NetCore.Component.DevelopmentTools	.NET Core development tools	16.4.29511.114	Required
Microsoft.NetCore.Component.SDK	.NET Core 3.1 SDK	16.4.29519.181	Required
Microsoft.NetCore.Component.Web	.NET Core development tools	16.4.29511.114	Required
Microsoft.VisualStudio.Component.Azure.AuthoringTools	Azure Authoring Tools	16.0.28625.61	Required
Microsoft.VisualStudio.Component.Azure.ClientLibs	Azure libraries for .NET	16.0.28315.86	Required
Microsoft.VisualStudio.Component.Azure.Compute.Emulator	Azure Compute Emulator	16.1.28810.153	Required
Microsoft.VisualStudio.Component.Azure.Storage.Emulator	Azure Storage Emulator	16.4.29313.120	Required
Microsoft.VisualStudio.Component.CloudExplorer	Cloud Explorer	16.0.28625.61	Required
Microsoft.VisualStudio.Component.Common.Azure.Tools	Connectivity and publishing tools	16.4.29409.204	Required
Microsoft.VisualStudio.Component.DockerTools	Container development tools	16.4.29409.204	Required
Microsoft.VisualStudio.Component.FSharp	F# language support	16.0.28315.86	Required
Microsoft.VisualStudio.Component.FSharp.WebTemplates	F# language support for web projects	16.3.29207.166	Required
Microsoft.VisualStudio.Component.IISExpress	IIS Express	16.0.28315.86	Required
Microsoft.VisualStudio.Component.IntelliCode	IntelliCode	0.1	Required

Component ID	Name	Version	Dependency Type
Microsoft.VisualStudio.Component.JavaScript.Diagnostics	JavaScript diagnostics	16.0.28517.75	Required
Microsoft.VisualStudio.Component.JavaScript.TypeScript	JavaScript and TypeScript language support	16.4.29409.204	Required
Microsoft.VisualStudio.Component.ManagedDesktop.Core	Managed Desktop Workload Core	16.4.29318.151	Required
Microsoft.VisualStudio.Component.MSODBC.SQL	SQL Server ODBC Driver	16.0.28625.61	Required
Microsoft.VisualStudio.Component.MSSQL.CMDLnUtils	SQL Server Command Line Utilities	16.0.28707.177	Required
Microsoft.VisualStudio.Component.NuGet	NuGet package manager	16.1.28829.92	Required
Microsoft.VisualStudio.Component.Roslyn.Compiler	C# and Visual Basic Roslyn compilers	16.0.28714.129	Required
Microsoft.VisualStudio.Component.Roslyn.LanguageServices	C# and Visual Basic	16.4.29429.68	Required
Microsoft.VisualStudio.Component.SQL.ADAL	SQL ADAL runtime	16.0.28517.75	Required
Microsoft.VisualStudio.Component.SQL.CLR	CLR data types for SQL Server	16.0.28315.86	Required
Microsoft.VisualStudio.Component.SQL.DataSources	Data sources for SQL Server support	16.0.28315.86	Required
Microsoft.VisualStudio.Component.SQL.LocalDB.Runtime	SQL Server Express 2016 LocalDB	16.0.28625.61	Required
Microsoft.VisualStudio.Component.SQL.SSDT	SQL Server Data Tools	16.3.29207.166	Required
Microsoft.VisualStudio.Component.TextTemplating	Text Template Transformation	16.0.28625.61	Required
Microsoft.VisualStudio.Component.TypeScript.3.7	TypeScript 3.7 SDK	16.0.29429.68	Required
Microsoft.VisualStudio.Component.Web	ASP.NET and web development tools	16.0.28517.75	Required
Microsoft.VisualStudio.ComponentGroup.Azure.Prerequisites	Azure development prerequisites	16.4.29409.204	Required

Component ID	Name	Version	Dependency Type
Microsoft.VisualStudio.ComponentGroup.AzureFunctions	Azure WebJobs Tools	16.0.28621.142	Required
Microsoft.VisualStudio.ComponentGroup.Web	ASP.NET and web development tools prerequisites	16.4.29318.151	Required
Microsoft.VisualStudio.ComponentGroup.WebToolsExtensions	ASP.NET and web development	16.0.28621.142	Required
Microsoft.Component.Azure.DataLake.Tools	Azure Data Lake and Stream Analytics Tools	16.4.29313.120	Recommended
Microsoft.Net.Component.4.5.1.TargetingPack	.NET Framework 4.5.1 targeting pack	16.0.28517.75	Recommended
Microsoft.Net.Component.4.6.TargetingPack	.NET Framework 4.6 targeting pack	16.0.28517.75	Recommended
Microsoft.Net.Component.4.TargetingPack	.NET Framework 4 targeting pack	16.0.28517.75	Recommended
Microsoft.Net.ComponentGroup.TargetingPacks.Common	.NET Framework 4 – 4.6 development tools	16.0.28516.191	Recommended
Microsoft.Net.Core.Component.SDK.2.1	.NET Core 2.1 LTS Runtime	16.4.29519.181	Recommended
Microsoft.VisualStudio.Component.AspNet45	Advanced ASP.NET features	16.0.28315.86	Recommended
Microsoft.VisualStudio.Component.Azure.Kubernetes.Tools	Visual Studio Tools for Kubernetes	16.0.28625.61	Recommended
Microsoft.VisualStudio.Component.Azure.PowerShell	Azure Powershell	16.4.29429.68	Recommended
Microsoft.VisualStudio.Component.Azure.ResourceManager.Tools	Azure Resource Manager core tools	16.4.29409.204	Recommended
Microsoft.VisualStudio.Component.Azure.ServiceFabric.Tools	Service Fabric Tools	16.4.29313.120	Recommended
Microsoft.VisualStudio.Component.Azure.Waverton	Azure Cloud Services core tools	16.4.29409.204	Recommended
Microsoft.VisualStudio.Component.Azure.Waverton.BuildTools	Azure Cloud Services build tools	16.3.29207.166	Recommended

Component ID	Name	Version	Dependency Type
Microsoft.VisualStudio.Component.Debugger.Snapshot	Snapshot Debugger	16.2.29003.222	Recommended
Microsoft.VisualStudio.Component.Debugger.TimeTravel	Time Travel Debugger	16.4.29511.114	Recommended
Microsoft.VisualStudio.Component.DiagnosticTools	.NET profiling tools	16.4.29429.68	Recommended
Microsoft.VisualStudio.Component.IntelliTrace.FrontEnd	IntelliTrace	16.4.29429.68	Recommended
Microsoft.VisualStudio.Component.WebDeploy	Web Deploy	16.0.28517.75	Recommended
Microsoft.VisualStudio.ComponentGroup.Azure.CloudServices	Azure Cloud Services tools	16.4.29409.204	Recommended
Microsoft.VisualStudio.ComponentGroup.Azure.ResourceManager.Tools	Azure Resource Manager tools	16.0.28528.71	Recommended
Microsoft.Net.Component.4.6.2.TargetingPack	.NET Framework 4.6.2 targeting pack	16.0.28517.75	Optional
Microsoft.Net.Component.4.7.1.TargetingPack	.NET Framework 4.7.1 targeting pack	16.0.28517.75	Optional
Microsoft.Net.Component.4.7.TargetingPack	.NET Framework 4.7 targeting pack	16.0.28517.75	Optional
Microsoft.Net.Component.4.8.TargetingPack	.NET Framework 4.8 targeting pack	16.4.29313.120	Optional
Microsoft.Net.ComponentGroup.4.6.1.DeveloperTools	.NET Framework 4.6.1 development tools	16.3.29207.166	Optional
Microsoft.Net.ComponentGroup.4.6.2.DeveloperTools	.NET Framework 4.6.2 development tools	16.3.29207.166	Optional
Microsoft.Net.ComponentGroup.4.7.1.DeveloperTools	.NET Framework 4.7.1 development tools	16.3.29207.166	Optional
Microsoft.Net.ComponentGroup.4.7.DeveloperTools	.NET Framework 4.7 development tools	16.3.29207.166	Optional
Microsoft.Net.ComponentGroup.4.8.DeveloperTools	.NET Framework 4.8 development tools	16.4.29318.151	Optional
Microsoft.VisualStudio.Component.Azure.Storage.AzCopy	Azure Storage AzCopy	16.0.28517.75	Optional

Component ID	Name	Version	Dependency Type
Microsoft.VisualStudio.Component.Wcf.Tooling	Windows Communication Foundation	16.0.28625.61	Optional

Data storage and processing

ID: Microsoft.VisualStudio.Workload.Data

Description: Connect, develop, and test data solutions with SQL Server, Azure Data Lake, or Hadoop.

Components included by this workload

Component ID	Name	Version	Dependency Type
Component.Microsoft.VisualStudio.RazorExtension	Razor Language Services	16.0.28714.129	Recommended
Component.Microsoft.Web.LibraryManager	Library Manager	16.0.28315.86	Recommended
Microsoft.Component.Azure.DataLake.Tools	Azure Data Lake and Stream Analytics Tools	16.4.29313.120	Recommended
Microsoft.Component.MSBuild	MSBuild	16.4.29429.68	Recommended
Microsoft.Net.Component.4.5.1.TargetingPack	.NET Framework 4.5.1 targeting pack	16.0.28517.75	Recommended
Microsoft.Net.Component.4.5.2.TargetingPack	.NET Framework 4.5.2 targeting pack	16.0.28517.75	Recommended
Microsoft.Net.Component.4.5.TargetingPack	.NET Framework 4.5 targeting pack	16.0.28517.75	Recommended
Microsoft.Net.Component.4.6.TargetingPack	.NET Framework 4.6 targeting pack	16.0.28517.75	Recommended
Microsoft.Net.Component.4.7.2.TargetingPack	.NET Framework 4.7.2 targeting pack	16.0.28517.75	Recommended
Microsoft.Net.Component.4.8.SDK	.NET Framework 4.8 SDK	16.4.29313.120	Recommended
Microsoft.Net.Component.4.TargetingPack	.NET Framework 4 targeting pack	16.0.28517.75	Recommended
Microsoft.Net.ComponentGroup.DevelopmentPrerequisites	.NET Framework 4.7.2 development tools	16.3.29207.166	Recommended
Microsoft.Net.ComponentGroup.TargetingPacks.Common	.NET Framework 4 – 4.6 development tools	16.0.28516.191	Recommended

Component ID	Name	Version	Dependency Type
Microsoft.NetCore.Component.SDK	.NET Core 3.1 SDK	16.4.29519.181	Recommended
Microsoft.VisualStudio.Component.Azure.AuthoringTools	Azure Authoring Tools	16.0.28625.61	Recommended
Microsoft.VisualStudio.Component.Azure.ClientLibs	Azure libraries for .NET	16.0.28315.86	Recommended
Microsoft.VisualStudio.Component.Azure.Compute.Emulator	Azure Compute Emulator	16.1.28810.153	Recommended
Microsoft.VisualStudio.Component.Azure.Storage.Emulator	Azure Storage Emulator	16.4.29313.120	Recommended
Microsoft.VisualStudio.Component.Azure.Waverton	Azure Cloud Services core tools	16.4.29409.204	Recommended
Microsoft.VisualStudio.Component.Azure.Waverton.BuildTools	Azure Cloud Services build tools	16.3.29207.166	Recommended
Microsoft.VisualStudio.Component.CloudExplorer	Cloud Explorer	16.0.28625.61	Recommended
Microsoft.VisualStudio.Component.Common.Azure.Tools	Connectivity and publishing tools	16.4.29409.204	Recommended
Microsoft.VisualStudio.Component.DockerTools	Container development tools	16.4.29409.204	Recommended
Microsoft.VisualStudio.Component.IISExpress	IIS Express	16.0.28315.86	Recommended
Microsoft.VisualStudio.Component.JavaScript.Diagnostics	JavaScript diagnostics	16.0.28517.75	Recommended
Microsoft.VisualStudio.Component.JavaScript.TypeScript	JavaScript and TypeScript language support	16.4.29409.204	Recommended
Microsoft.VisualStudio.Component.ManagedDesktop.Core	Managed Desktop Workload Core	16.4.29318.151	Recommended
Microsoft.VisualStudio.Component.MSODBC.SQL	SQL Server ODBC Driver	16.0.28625.61	Recommended
Microsoft.VisualStudio.Component.MSSQL.CMDLnUtils	SQL Server Command Line Utilities	16.0.28707.177	Recommended
Microsoft.VisualStudio.Component.NuGet	NuGet package manager	16.1.28829.92	Recommended

Component ID	Name	Version	Dependency Type
Microsoft.VisualStudio.Component.Roslyn.Compiler	C# and Visual Basic Roslyn compilers	16.0.28714.129	Recommended
Microsoft.VisualStudio.Component.Roslyn.LanguageServices	C# and Visual Basic	16.4.29429.68	Recommended
Microsoft.VisualStudio.Component.SQL.ADAL	SQL ADAL runtime	16.0.28517.75	Recommended
Microsoft.VisualStudio.Component.SQL.CLR	CLR data types for SQL Server	16.0.28315.86	Recommended
Microsoft.VisualStudio.Component.SQL.DataSources	Data sources for SQL Server support	16.0.28315.86	Recommended
Microsoft.VisualStudio.Component.SQL.LocalDB.Runtime	SQL Server Express 2016 LocalDB	16.0.28625.61	Recommended
Microsoft.VisualStudio.Component.SQL.SSDT	SQL Server Data Tools	16.3.29207.166	Recommended
Microsoft.VisualStudio.Component.TextTemplating	Text Template Transformation	16.0.28625.61	Recommended
Microsoft.VisualStudio.Component.TypeScript.3.7	TypeScript 3.7 SDK	16.0.29429.68	Recommended
Microsoft.VisualStudio.Component.Web	ASP.NET and web development tools	16.0.28517.75	Recommended
Microsoft.VisualStudio.ComponentGroup.Web	ASP.NET and web development tools prerequisites	16.4.29318.151	Recommended
Microsoft.VisualStudio.ComponentGroup.WebToolsExtensions	ASP.NET and web development	16.0.28621.142	Recommended
Microsoft.Net.Component.4.6.1.TargetingPack	.NET Framework 4.6.1 targeting pack	16.0.28517.75	Optional
Microsoft.VisualStudio.Component.FSharp.Desktop	F# desktop language support	16.0.28315.86	Optional

Data science and analytical applications

ID: Microsoft.VisualStudio.Workload.DataScience

Description: Languages and tooling for creating data science applications, including Python and F#.

Components included by this workload

Component ID	Name	Version	Dependency Type
Microsoft.Component.PythonTools	Python language support	16.4.29429.68	Recommended
Microsoft.Component.PythonTools.Minicondax64	Python miniconda	16.2.29003.222	Recommended
Microsoft.Component.PythonTools.Web	Python web support	16.0.28517.75	Recommended
Microsoft.Net.Component.4.6.1.TargetingPack	.NET Framework 4.6.1 targeting pack	16.0.28517.75	Recommended
Microsoft.VisualStudio.Component.Common.Azure.Tools	Connectivity and publishing tools	16.4.29409.204	Recommended
Microsoft.VisualStudio.Component.FSharp.Desktop	F# desktop language support	16.0.28315.86	Recommended
Microsoft.VisualStudio.Component.JavaScript.TypeScript	JavaScript and TypeScript language support	16.4.29409.204	Recommended
Microsoft.VisualStudio.Component.NuGet	NuGet package manager	16.1.28829.92	Recommended
Microsoft.VisualStudio.Component.Roslyn.Compiler	C# and Visual Basic Roslyn compilers	16.0.28714.129	Recommended
Microsoft.VisualStudio.Component.Roslyn.LanguageServices	C# and Visual Basic	16.4.29429.68	Recommended
Microsoft.VisualStudio.Component.TypeScript.3.7	TypeScript 3.7 SDK	16.0.29429.68	Recommended
Microsoft.VisualStudio.Component.WebDeploy	Web Deploy	16.0.28517.75	Recommended
Microsoft.VisualStudio.ComponentGroup.WebToolsExtensions	ASP.NET and web development	16.0.28621.142	Recommended
Microsoft.ComponentGroup.PythonTools.NativeDevelopment	Python native development tools	16.2.29020.229	Optional
Microsoft.VisualStudio.Component.Graphics.Tools	Graphics debugger and GPU profiler for DirectX	16.0.28625.61	Optional
Microsoft.VisualStudio.Component.VC.Corelde	C++ core features	16.0.28625.61	Optional
Microsoft.VisualStudio.Component.VC.DiagnosticTools	C++ profiling tools	16.4.29429.68	Optional

Component ID	Name	Version	Dependency Type
Microsoft.VisualStudio.Component.VC.Tools.x86.x64	MSVC v142 - VS 2019 C++ x64/x86 build tools (v14.24)	16.4.29409.204	Optional
Microsoft.VisualStudio.Component.Windows10SDK	Windows Universal C Runtime	16.4.29409.204	Optional
Microsoft.VisualStudio.Component.Windows10SDK.18362	Windows 10 SDK (10.0.18362.0)	16.1.28829.92	Optional

.NET desktop development

ID: Microsoft.VisualStudio.Workload.ManagedDesktop

Description: Build WPF, Windows Forms, and console applications using C#, Visual Basic, and F# with .NET Core and .NET Framework.

Components included by this workload

Component ID	Name	Version	Dependency Type
Microsoft.Component.MSBuild	MSBuild	16.4.29429.68	Required
Microsoft.Net.Component.4.7.2.TargetingPack	.NET Framework 4.7.2 targeting pack	16.0.28517.75	Required
Microsoft.Net.Component.4.8.SDK	.NET Framework 4.8 SDK	16.4.29313.120	Required
Microsoft.Net.ComponentGroup.DevelopmentPrerequisites	.NET Framework 4.7.2 development tools	16.3.29207.166	Required
Microsoft.NetCore.Component.SDK	.NET Core 3.1 SDK	16.4.29519.181	Required
Microsoft.VisualStudio.Component.IntelliCode	IntelliCode	0.1	Required
Microsoft.VisualStudio.Component.ManagedDesktop.Core	Managed Desktop Workload Core	16.4.29318.151	Required
Microsoft.VisualStudio.Component.ManagedDesktop.Prerequisites	.NET desktop development tools	16.4.29429.68	Required
Microsoft.VisualStudio.Component.Roslyn.Compiler	C# and Visual Basic Roslyn compilers	16.0.28714.129	Required
Microsoft.VisualStudio.Component.Roslyn.LanguageServices	C# and Visual Basic	16.4.29429.68	Required

Component ID	Name	Version	Dependency Type
Microsoft.VisualStudio.Component.SQL CLR	CLR data types for SQL Server	16.0.28315.86	Required
Microsoft.VisualStudio.Component.TextTemplating	Text Template Transformation	16.0.28625.61	Required
Component.Microsoft.VisualStudio.LiveShare	Live Share	1.0.1076	Recommended
Microsoft.ComponentGroup.Blend	Blend for Visual Studio	16.0.28315.86	Recommended
Microsoft.Net.Component.4.5.1.TargetingPack	.NET Framework 4.5.1 targeting pack	16.0.28517.75	Recommended
Microsoft.Net.Component.4.5.2.TargetingPack	.NET Framework 4.5.2 targeting pack	16.0.28517.75	Recommended
Microsoft.Net.Component.4.5.TargetingPack	.NET Framework 4.5 targeting pack	16.0.28517.75	Recommended
Microsoft.Net.Component.4.6.1.TargetingPack	.NET Framework 4.6.1 targeting pack	16.0.28517.75	Recommended
Microsoft.Net.Component.4.6.TargetingPack	.NET Framework 4.6 targeting pack	16.0.28517.75	Recommended
Microsoft.Net.Component.4.TargetingPack	.NET Framework 4 targeting pack	16.0.28517.75	Recommended
Microsoft.Net.ComponentGroup.TargetingPacks.Common	.NET Framework 4 – 4.6 development tools	16.0.28516.191	Recommended
Microsoft.Net.Core.Component.SDK.2.1	.NET Core 2.1 LTS Runtime	16.4.29519.181	Recommended
Microsoft.NetCore.Component.DevelopmentTools	.NET Core development tools	16.4.29511.114	Recommended
Microsoft.VisualStudio.Component.Debugger.JustInTime	Just-In-Time debugger	16.0.28517.75	Recommended
Microsoft.VisualStudio.Component.DiagnosticTools	.NET profiling tools	16.4.29429.68	Recommended
Microsoft.VisualStudio.Component.EntityFramework	Entity Framework 6 tools	16.0.28315.86	Recommended
Microsoft.VisualStudio.Component.FSharp	F# language support	16.0.28315.86	Recommended
Microsoft.VisualStudio.Component.IntelliTrace.FrontEnd	IntelliTrace	16.4.29429.68	Recommended

Component ID	Name	Version	Dependency Type
Microsoft.VisualStudio.Component.LiveUnitTesting	Live Unit Testing	16.1.28811.260	Recommended
Microsoft.VisualStudio.Component.NuGet	NuGet package manager	16.1.28829.92	Recommended
Component.Dotfuscator	PreEmptive Protection - Dotfuscator	16.0.28528.71	Optional
Component.Microsoft.VisualStudio.RazorExtension	Razor Language Services	16.0.28714.129	Optional
Component.Microsoft.Web.LibraryManager	Library Manager	16.0.28315.86	Optional
Microsoft.Net.Component.4.6.2.TargetingPack	.NET Framework 4.6.2 targeting pack	16.0.28517.75	Optional
Microsoft.Net.Component.4.7.1.TargetingPack	.NET Framework 4.7.1 targeting pack	16.0.28517.75	Optional
Microsoft.Net.Component.4.7.TargetingPack	.NET Framework 4.7 targeting pack	16.0.28517.75	Optional
Microsoft.Net.Component.4.8.TargetingPack	.NET Framework 4.8 targeting pack	16.4.29313.120	Optional
Microsoft.Net.ComponentGroup.4.6.1.DeveloperTools	.NET Framework 4.6.1 development tools	16.3.29207.166	Optional
Microsoft.Net.ComponentGroup.4.6.2.DeveloperTools	.NET Framework 4.6.2 development tools	16.3.29207.166	Optional
Microsoft.Net.ComponentGroup.4.7.1.DeveloperTools	.NET Framework 4.7.1 development tools	16.3.29207.166	Optional
Microsoft.Net.ComponentGroup.4.7.DeveloperTools	.NET Framework 4.7 development tools	16.3.29207.166	Optional
Microsoft.Net.ComponentGroup.4.8.DeveloperTools	.NET Framework 4.8 development tools	16.4.29318.151	Optional
Microsoft.VisualStudio.Component.ClassDesigner	Class Designer	16.0.28528.71	Optional
Microsoft.VisualStudio.Component.CodeClone	Code Clone	16.4.29409.204	Optional
Microsoft.VisualStudio.Component.CodeMap	Code Map	16.0.28625.61	Optional
Microsoft.VisualStudio.Component.Common.Azure.Tools	Connectivity and publishing tools	16.4.29409.204	Optional

Component ID	Name	Version	Dependency Type
Microsoft.VisualStudio.Component.DependencyValidation.Enterprise	Live Dependency Validation	16.0.28625.61	Optional
Microsoft.VisualStudio.Component.DockerTools	Container development tools	16.4.29409.204	Optional
Microsoft.VisualStudio.Component.FSharp.Desktop	F# desktop language support	16.0.28315.86	Optional
Microsoft.VisualStudio.Component.GraphDocument	DGML editor	16.0.28625.61	Optional
Microsoft.VisualStudio.Component.IIExpress	IIS Express	16.0.28315.86	Optional
Microsoft.VisualStudio.Component.JavaScript.Diagnostics	JavaScript diagnostics	16.0.28517.75	Optional
Microsoft.VisualStudio.Component.JavaScript.TypeScript	JavaScript and TypeScript language support	16.4.29409.204	Optional
Microsoft.VisualStudio.Component.MSODBC.SQL	SQL Server ODBC Driver	16.0.28625.61	Optional
Microsoft.VisualStudio.Component.MSSQL.CMDLnUtils	SQL Server Command Line Utilities	16.0.28707.177	Optional
Microsoft.VisualStudio.Component.PortableLibrary	.NET Portable Library targeting pack	16.0.28517.75	Optional
Microsoft.VisualStudio.Component.SQL.ADAL	SQL ADAL runtime	16.0.28517.75	Optional
Microsoft.VisualStudio.Component.SQL.DataSources	Data sources for SQL Server support	16.0.28315.86	Optional
Microsoft.VisualStudio.Component.SQL.LocalDB.Runtime	SQL Server Express 2016 LocalDB	16.0.28625.61	Optional
Microsoft.VisualStudio.Component.SQL.SSDT	SQL Server Data Tools	16.3.29207.166	Optional
Microsoft.VisualStudio.Component.TypeScript.3.7	TypeScript 3.7 SDK	16.0.29429.68	Optional
Microsoft.VisualStudio.Component.Wcf.Tooling	Windows Communication Foundation	16.0.28625.61	Optional
Microsoft.VisualStudio.Component.Web	ASP.NET and web development tools	16.0.28517.75	Optional

Component ID	Name	Version	Dependency Type
Microsoft.VisualStudio.Component.Windows10SDK.18362	Windows 10 SDK (10.0.18362.0)	16.1.28829.92	Optional
Microsoft.VisualStudio.ComponentGroup.ArchitectureTools.Managed	Architecture and analysis tools	16.4.29429.68	Optional
Microsoft.VisualStudio.ComponentGroup.MSIX.Packaging	MSIX Packaging Tools	16.4.29409.204	Optional
Microsoft.VisualStudio.ComponentGroup.Web	ASP.NET and web development tools prerequisites	16.4.29318.151	Optional
Microsoft.VisualStudio.ComponentGroup.WebToolsExtensions	ASP.NET and web development	16.0.28621.142	Optional

Game development with Unity

ID: Microsoft.VisualStudio.Workload.ManagedGame

Description: Create 2D and 3D games with Unity, a powerful cross-platform development environment.

Components included by this workload

Component ID	Name	Version	Dependency Type
Microsoft.Net.Component.3.5.DeveloperTools	.NET Framework 3.5 development tools	16.0.28517.75	Required
Microsoft.Net.Component.4.7.1.TargetingPack	.NET Framework 4.7.1 targeting pack	16.0.28517.75	Required
Microsoft.VisualStudio.Component.NuGet	NuGet package manager	16.1.28829.92	Required
Microsoft.VisualStudio.Component.Roslyn.Compiler	C# and Visual Basic Roslyn compilers	16.0.28714.129	Required
Microsoft.VisualStudio.Component.Roslyn.LanguageServices	C# and Visual Basic	16.4.29429.68	Required
Microsoft.VisualStudio.Component.Unity	Visual Studio Tools for Unity	16.0.28315.86	Required
Component.UnityEngine.x64	Unity 2019.2 64-bit Editor	16.4.29429.68	Recommended
Component.UnityEngine.x86	Unity 5.6 32-bit Editor	16.1.28811.260	Recommended

Linux development with C++

ID: Microsoft.VisualStudio.Workload.NativeCrossPlat

Description: Create and debug applications running in a Linux environment.

Components included by this workload

COMPONENT ID	NAME	VERSION	DEPENDENCY TYPE
Component.MDD.Linux	C++ for Linux Development	16.4.29511.114	Required
Microsoft.VisualStudio.Component.IntelliCode	IntelliCode	0.1	Required
Microsoft.VisualStudio.Component.VC.CoreLde	C++ core features	16.0.28625.61	Required
Component.Linux.CMake	C++ CMake tools for Linux	16.2.29003.222	Recommended
Microsoft.VisualStudio.ComponentGroup.WebToolsExtensions	ASP.NET and web development	16.0.28621.142	Recommended
Component.MDD.Linux.GCC.arm	Embedded and IoT development tools	16.4.29429.68	Optional

Desktop development with C++

ID: Microsoft.VisualStudio.Workload.NativeDesktop

Description: Build modern C++ apps for Windows using tools of your choice, including MSVC, Clang, CMake, or MSBuild.

Components included by this workload

COMPONENT ID	NAME	VERSION	DEPENDENCY TYPE
Microsoft.Component.MSBuild	MSBuild	16.4.29429.68	Required
Microsoft.VisualStudio.Component.ClassDesigner	Class Designer	16.0.28528.71	Required
Microsoft.VisualStudio.Component.CodeMap	Code Map	16.0.28625.61	Required
Microsoft.VisualStudio.Component.GraphDocument	DGML editor	16.0.28625.61	Required
Microsoft.VisualStudio.Component.IntelliCode	IntelliCode	0.1	Required
Microsoft.VisualStudio.Component.Roslyn.Compiler	C# and Visual Basic Roslyn compilers	16.0.28714.129	Required
Microsoft.VisualStudio.Component.SQL.LocalDB.Runtime	SQL Server Express 2016 LocalDB	16.0.28625.61	Required

Component ID	Name	Version	Dependency Type
Microsoft.VisualStudio.Component.TextTemplating	Text Template Transformation	16.0.28625.61	Required
Microsoft.VisualStudio.Component.VC.CoreRedist	C++ core features	16.0.28625.61	Required
Microsoft.VisualStudio.Component.VC.Redist.14.Latest	C++ 2019 Redistributable Update	16.4.29429.68	Required
Microsoft.VisualStudio.ComponentGroup.ArchitectureTools.Native	Architecture tools for C++	16.0.28621.142	Required
Microsoft.VisualStudio.ComponentGroup.NativeDesktop.Core	C++ core desktop features	16.2.29012.281	Required
Component.Microsoft.VisualStudio.LiveShare	Live Share	1.0.1076	Recommended
Microsoft.VisualStudio.Component.Debugger.JustInTime	Just-In-Time debugger	16.0.28517.75	Recommended
Microsoft.VisualStudio.Component.Graphics.Tools	Graphics debugger and GPU profiler for DirectX	16.0.28625.61	Recommended
Microsoft.VisualStudio.Component.IntelliTrace.FrontEnd	IntelliTrace	16.4.29429.68	Recommended
Microsoft.VisualStudio.Component.NuGet	NuGet package manager	16.1.28829.92	Recommended
Microsoft.VisualStudio.Component.VC.ASAN	C++ AddressSanitizer (Experimental)	16.4.29429.68	Recommended
Microsoft.VisualStudio.Component.VC.ATL	C++ ATL for latest v142 build tools (x86 & x64)	16.4.29313.120	Recommended
Microsoft.VisualStudio.Component.VC.CMake.Project	C++ CMake tools for Windows	16.3.29103.31	Recommended
Microsoft.VisualStudio.Component.VC.DiagnosticTools	C++ profiling tools	16.4.29429.68	Recommended
Microsoft.VisualStudio.Component.VC.TestAdapterForBoostTest	Test Adapter for Boost.Test	16.0.28517.75	Recommended
Microsoft.VisualStudio.Component.VC.TestAdapterForGoogleTest	Test Adapter for Google Test	16.0.28517.75	Recommended
Microsoft.VisualStudio.Component.VC.Tools.x86.x64	MSVC v142 - VS 2019 C++ x64/x86 build tools (v14.24)	16.4.29409.204	Recommended

Component ID	Name	Version	Dependency Type
Microsoft.VisualStudio.Component.Windows10SDK.18362	Windows 10 SDK (10.0.18362.0)	16.1.28829.92	Recommended
Microsoft.VisualStudio.ComponentGroup.WebToolsExtensions.CMake	JSON editor	16.3.29207.166	Recommended
Component.Incredibuild	Incredibuild - Build Acceleration	16.0.28528.71	Optional
Component.IncredibuildMenu	IncredibuildMenu	1.5.0.10	Optional
Microsoft.Component.VC.Runtime.UCRTSDK	Windows Universal CRT SDK	16.0.28625.61	Optional
Microsoft.Net.Component.4.6.1.TargetingPack	.NET Framework 4.6.1 targeting pack	16.0.28517.75	Optional
Microsoft.Net.Component.4.8.SDK	.NET Framework 4.8 SDK	16.4.29313.120	Optional
Microsoft.VisualStudio.Component.JavaScript.Diagnostics	JavaScript diagnostics	16.0.28517.75	Optional
Microsoft.VisualStudio.Component.VC.140	MSVC v140 - VS 2015 C++ build tools (v14.00)	16.0.28625.61	Optional
Microsoft.VisualStudio.Component.VC.ATLMFC	C++ MFC for latest v142 build tools (x86 & x64)	16.4.29313.120	Optional
Microsoft.VisualStudio.Component.VC.CLI.Support	C++/CLI support for v142 build tools (14.24)	16.4.29409.204	Optional
Microsoft.VisualStudio.Component.VC.Llvm.Clang	C++ Clang Compiler for Windows (9.0.0)	16.4.29511.114	Optional
Microsoft.VisualStudio.Component.VC.Llvm.ClangToolset	C++ Clang-cl for v142 build tools (x64/x86)	16.3.29207.166	Optional
Microsoft.VisualStudio.Component.VC.Modules.x86.x64	C++ Modules for v142 build tools (x64/x86 – experimental)	16.0.28625.61	Optional
Microsoft.VisualStudio.Component.VC.v141.x86.x64	MSVC v141 - VS 2017 C++ x64/x86 build tools (v14.16)	16.1.28829.92	Optional
Microsoft.VisualStudio.Component.Windows10SDK.16299	Windows 10 SDK (10.0.16299.0)	16.0.28517.75	Optional

Component ID	Name	Version	Dependency Type
Microsoft.VisualStudio.Component.Windows10SDK.17134	Windows 10 SDK (10.0.17134.0)	16.0.28517.75	Optional
Microsoft.VisualStudio.Component.Windows10SDK.17763	Windows 10 SDK (10.0.17763.0)	16.0.28517.75	Optional
Microsoft.VisualStudio.ComponentGroup.NativeDesktop.Llvm.Clang	C++ Clang tools for Windows (9.0.0 - x64/x86)	16.4.29511.114	Optional

Game development with C++

ID: Microsoft.VisualStudio.Workload.NativeGame

Description: Use the full power of C++ to build professional games powered by DirectX, Unreal, or Cocos2d.

Components included by this workload

Component ID	Name	Version	Dependency Type
Microsoft.VisualStudio.Component.IntelliCode	IntelliCode	0.1	Required
Microsoft.VisualStudio.Component.VC.CoreLDE	C++ core features	16.0.28625.61	Required
Microsoft.VisualStudio.Component.VC.Redist.14.Latest	C++ 2019 Redistributable Update	16.4.29429.68	Required
Microsoft.VisualStudio.Component.VC.Tools.x86.x64	MSVC v142 - VS 2019 C++ x64/x86 build tools (v14.24)	16.4.29409.204	Required
Microsoft.VisualStudio.Component.Windows10SDK	Windows Universal C Runtime	16.4.29409.204	Required
Microsoft.VisualStudio.Component.Graphics.Tools	Graphics debugger and GPU profiler for DirectX	16.0.28625.61	Recommended
Microsoft.VisualStudio.Component.IntelliTrace.FrontEnd	IntelliTrace	16.4.29429.68	Recommended
Microsoft.VisualStudio.Component.VC.ASAN	C++ AddressSanitizer (Experimental)	16.4.29429.68	Recommended
Microsoft.VisualStudio.Component.VC.DiagnosticTools	C++ profiling tools	16.4.29429.68	Recommended
Microsoft.VisualStudio.Component.Windows10SDK.18362.2	Windows 10 SDK (10.0.18362.0)	16.1.28829.92	Recommended

Component ID	Name	Version	Dependency Type
Component.Android.NDK.R16B	Android NDK (R16B)	16.4.29519.181	Optional
Component.Android.SDK25.Private	Android SDK setup (API level 25) (local install for Mobile development with C++)	16.0.28625.61	Optional
Component.Ant	Apache Ant (1.9.3)	1.9.3.8	Optional
Component.Cocos	Cocos	16.0.28315.86	Optional
Component.Incredibuild	Incredibuild - Build Acceleration	16.0.28528.71	Optional
Component.IncredibuildMenu	IncredibuildMenu	1.5.0.10	Optional
Component.MDD.Android	C++ Android development tools	16.0.28517.75	Optional
Component.OpenJDK	OpenJDK (Microsoft distribution)	16.1.28811.260	Optional
Component.Unreal	Unreal Engine installer	16.1.28810.153	Optional
Component.Unreal.Android	Android IDE support for Unreal engine	16.1.28810.153	Optional
Microsoft.Net.Component.4.5.1.TargetingPack	.NET Framework 4.5.1 targeting pack	16.0.28517.75	Optional
Microsoft.Net.Component.4.5.2.TargetingPack	.NET Framework 4.5.2 targeting pack	16.0.28517.75	Optional
Microsoft.Net.Component.4.5.TargetingPack	.NET Framework 4.5 targeting pack	16.0.28517.75	Optional
Microsoft.Net.Component.4.6.2.TargetingPack	.NET Framework 4.6.2 targeting pack	16.0.28517.75	Optional
Microsoft.Net.Component.4.6.TargetingPack	.NET Framework 4.6 targeting pack	16.0.28517.75	Optional
Microsoft.Net.Component.4.7.2.TargetingPack	.NET Framework 4.7.2 targeting pack	16.0.28517.75	Optional
Microsoft.Net.Component.4.8.SDK	.NET Framework 4.8 SDK	16.4.29313.120	Optional
Microsoft.Net.Component.4.TargetingPack	.NET Framework 4 targeting pack	16.0.28517.75	Optional

Component ID	Name	Version	Dependency Type
Microsoft.Net.ComponentGroup.DevelopmentPrerequisites	.NET Framework 4.7.2 development tools	16.3.29207.166	Optional
Microsoft.Net.ComponentGroup.TargetingPacks.Common	.NET Framework 4 – 4.6 development tools	16.0.28516.191	Optional
Microsoft.VisualStudio.Component.NuGet.BuildTools	NuGet targets and build tasks	16.1.28829.92	Optional
Microsoft.VisualStudio.Component.Roslyn.Compiler	C# and Visual Basic Roslyn compilers	16.0.28714.129	Optional
Microsoft.VisualStudio.Component.Roslyn.LanguageServices	C# and Visual Basic	16.4.29429.68	Optional
Microsoft.VisualStudio.Component.Windows10SDK.16299	Windows 10 SDK (10.0.16299.0)	16.0.28517.75	Optional
Microsoft.VisualStudio.Component.Windows10SDK.17134	Windows 10 SDK (10.0.17134.0)	16.0.28517.75	Optional
Microsoft.VisualStudio.Component.Windows10SDK.17763	Windows 10 SDK (10.0.17763.0)	16.0.28517.75	Optional

Mobile development with C++

ID: Microsoft.VisualStudio.Workload.NativeMobile

Description: Build cross-platform applications for iOS, Android or Windows using C++.

Components included by this workload

Component ID	Name	Version	Dependency Type
Component.Android.SDK25.Private	Android SDK setup (API level 25) (local install for Mobile development with C++)	16.0.28625.61	Required
Component.OpenJDK	OpenJDK (Microsoft distribution)	16.1.28811.260	Required
Microsoft.VisualStudio.Component.IntelliCode	IntelliCode	0.1	Required
Microsoft.VisualStudio.Component.VC.Corelde	C++ core features	16.0.28625.61	Required
Component.Android.NDK.R16B	Android NDK (R16B)	16.4.29519.181	Recommended

COMPONENT ID	NAME	VERSION	DEPENDENCY TYPE
Component.Ant	Apache Ant (1.9.3)	1.9.3.8	Recommended
Component.MDD.Android	C++ Android development tools	16.0.28517.75	Recommended
Component.Android.NDK.R16B_3264	Android NDK (R16B) (32bit)	16.4.29519.181	Optional
Component.Google.Android.Emulator.API25.Private	Google Android Emulator (API Level 25) (local install)	16.1.28810.153	Optional
Component.HAXM.Private	Intel Hardware Accelerated Execution Manager (HAXM) (local install)	16.0.28528.71	Optional
Component.Incredibuild	Incredibuild - Build Acceleration	16.0.28528.71	Optional
Component.IncredibuildMenu	IncredibuildMenu	1.5.0.10	Optional
Component.MDD.iOS	C++ iOS development tools	16.0.28517.75	Optional

.NET Core cross-platform development

ID: Microsoft.VisualStudio.Workload.NetCoreTools

Description: Build cross-platform applications using .NET Core, ASP.NET Core, HTML/JavaScript, and Containers including Docker support.

Components included by this workload

COMPONENT ID	NAME	VERSION	DEPENDENCY TYPE
Component.Microsoft.VisualStudio.RazorExtension	Razor Language Services	16.0.28714.129	Required
Component.Microsoft.Web.LibraryManager	Library Manager	16.0.28315.86	Required
Microsoft.Component_MSBUILD	MSBuild	16.4.29429.68	Required
Microsoft.Net.Component.4.5.2.TargetingPack	.NET Framework 4.5.2 targeting pack	16.0.28517.75	Required
Microsoft.Net.Component.4.5.TargetingPack	.NET Framework 4.5 targeting pack	16.0.28517.75	Required
Microsoft.Net.Component.4.6.1.TargetingPack	.NET Framework 4.6.1 targeting pack	16.0.28517.75	Required

Component ID	Name	Version	Dependency Type
Microsoft.Net.Component.4.7.2.TargetingPack	.NET Framework 4.7.2 targeting pack	16.0.28517.75	Required
Microsoft.Net.Component.4.8.SDK	.NET Framework 4.8 SDK	16.4.29313.120	Required
Microsoft.Net.ComponentGroup.DevelopmentPrerequisites	.NET Framework 4.7.2 development tools	16.3.29207.166	Required
Microsoft.NetCore.Component.DevelopmentTools	.NET Core development tools	16.4.29511.114	Required
Microsoft.NetCore.Component.SDK	.NET Core 3.1 SDK	16.4.29519.181	Required
Microsoft.NetCore.Component.Web	.NET Core development tools	16.4.29511.114	Required
Microsoft.VisualStudio.Component.Common.Azure.Tools	Connectivity and publishing tools	16.4.29409.204	Required
Microsoft.VisualStudio.Component.DockerTools	Container development tools	16.4.29409.204	Required
Microsoft.VisualStudio.Component.FSharp	F# language support	16.0.28315.86	Required
Microsoft.VisualStudio.Component.FSharp.WebTemplates	F# language support for web projects	16.3.29207.166	Required
Microsoft.VisualStudio.Component.IISExpress	IIS Express	16.0.28315.86	Required
Microsoft.VisualStudio.Component.IntelliCode	IntelliCode	0.1	Required
Microsoft.VisualStudio.Component.JavaScript.Diagnostics	JavaScript diagnostics	16.0.28517.75	Required
Microsoft.VisualStudio.Component.JavaScript.TypeScript	JavaScript and TypeScript language support	16.4.29409.204	Required
Microsoft.VisualStudio.Component.ManagedDesktop.Core	Managed Desktop Workload Core	16.4.29318.151	Required
Microsoft.VisualStudio.Component.MSODBC.SQL	SQL Server ODBC Driver	16.0.28625.61	Required
Microsoft.VisualStudio.Component.MSSQL.CMDLnUtils	SQL Server Command Line Utilities	16.0.28707.177	Required

Component ID	Name	Version	Dependency Type
Microsoft.VisualStudio.Component.NuGet	NuGet package manager	16.1.28829.92	Required
Microsoft.VisualStudio.Component.Roslyn.Compiler	C# and Visual Basic Roslyn compilers	16.0.28714.129	Required
Microsoft.VisualStudio.Component.Roslyn.LanguageServices	C# and Visual Basic	16.4.29429.68	Required
Microsoft.VisualStudio.Component.SQL.ADAL	SQL ADAL runtime	16.0.28517.75	Required
Microsoft.VisualStudio.Component.SQL.CLR	CLR data types for SQL Server	16.0.28315.86	Required
Microsoft.VisualStudio.Component.SQL.DataSources	Data sources for SQL Server support	16.0.28315.86	Required
Microsoft.VisualStudio.Component.SQL.LocalDB.Runtime	SQL Server Express 2016 LocalDB	16.0.28625.61	Required
Microsoft.VisualStudio.Component.SQL.SSDT	SQL Server Data Tools	16.3.29207.166	Required
Microsoft.VisualStudio.Component.TextTemplating	Text Template Transformation	16.0.28625.61	Required
Microsoft.VisualStudio.Component.TypeScript.3.7	TypeScript 3.7 SDK	16.0.29429.68	Required
Microsoft.VisualStudio.ComponentGroup.Web	ASP.NET and web development tools prerequisites	16.4.29318.151	Required
Microsoft.VisualStudio.ComponentGroup.WebToolsExtensions	ASP.NET and web development	16.0.28621.142	Required
Component.Microsoft.VisualStudio.LiveShare	Live Share	1.0.1076	Recommended
Component.Microsoft.VisualStudio.Web.AzureFunctions	Azure WebJobs Tools	16.0.28714.129	Recommended
Microsoft.Net.Core.Component.SDK.2.1	.NET Core 2.1 LTS Runtime	16.4.29519.181	Recommended
Microsoft.VisualStudio.Component.AppInsights.Tools	Developer Analytics tools	16.4.29429.68	Recommended
Microsoft.VisualStudio.Component.Azure.AuthoringTools	Azure Authoring Tools	16.0.28625.61	Recommended

Component ID	Name	Version	Dependency Type
Microsoft.VisualStudio.Component.Azure.ClientLibs	Azure libraries for .NET	16.0.28315.86	Recommended
Microsoft.VisualStudio.Component.Azure.Compute.Emulator	Azure Compute Emulator	16.1.28810.153	Recommended
Microsoft.VisualStudio.Component.Azure.Storage.Emulator	Azure Storage Emulator	16.4.29313.120	Recommended
Microsoft.VisualStudio.Component.CloudExplorer	Cloud Explorer	16.0.28625.61	Recommended
Microsoft.VisualStudio.Component.Debugger.Snapshot	Snapshot Debugger	16.2.29003.222	Recommended
Microsoft.VisualStudio.Component.Debugger.TimeTravel	Time Travel Debugger	16.4.29511.114	Recommended
Microsoft.VisualStudio.Component.DiagnosticTools	.NET profiling tools	16.4.29429.68	Recommended
Microsoft.VisualStudio.Component.IntelliTrace.FrontEnd	IntelliTrace	16.4.29429.68	Recommended
Microsoft.VisualStudio.Component.LiveUnitTesting	Live Unit Testing	16.1.28811.260	Recommended
Microsoft.VisualStudio.Component.Web	ASP.NET and web development tools	16.0.28517.75	Recommended
Microsoft.VisualStudio.Component.WebDeploy	Web Deploy	16.0.28517.75	Recommended
Microsoft.VisualStudio.ComponentGroup.AzureFunctions	Azure WebJobs Tools	16.0.28621.142	Recommended
Microsoft.VisualStudio.ComponentGroup.Web.CloudTools	Cloud tools for web development	16.2.29003.222	Recommended
Microsoft.VisualStudio.Component.Windows10SDK.18362	Windows 10 SDK (10.0.18362.0)	16.1.28829.92	Optional
Microsoft.VisualStudio.ComponentGroup.IISDevelopment	Development time IIS support	16.0.28315.86	Optional
Microsoft.VisualStudio.ComponentGroup.MSIX.Packaging	MSIX Packaging Tools	16.4.29409.204	Optional

Mobile development with .NET

ID: Microsoft.VisualStudio.Workload.NetCrossPlat

Description: Build cross-platform applications for iOS, Android or Windows using Xamarin.

Components included by this workload

Component ID	Name	Version	Dependency Type
Component.OpenJDK	OpenJDK (Microsoft distribution)	16.1.28811.260	Required
Component.Xamarin	Xamarin	16.4.29409.204	Required
Component.Xamarin.RemotedSimulator	Xamarin Remoted Simulator	16.0.28315.86	Required
Microsoft.Component.MSBuild	MSBuild	16.4.29429.68	Required
Microsoft.Net.Component.4.6.1.TargetingPack	.NET Framework 4.6.1 targeting pack	16.0.28517.75	Required
Microsoft.Net.Component.4.7.2.TargetingPack	.NET Framework 4.7.2 targeting pack	16.0.28517.75	Required
Microsoft.Net.Component.4.8.SDK	.NET Framework 4.8 SDK	16.4.29313.120	Required
Microsoft.Net.ComponentGroup.DevelopmentPrerequisites	.NET Framework 4.7.2 development tools	16.3.29207.166	Required
Microsoft.NetCore.Component.DevelopmentTools	.NET Core development tools	16.4.29511.114	Required
Microsoft.NetCore.Component.SDK	.NET Core 3.1 SDK	16.4.29519.181	Required
Microsoft.VisualStudio.Component.FSharp	F# language support	16.0.28315.86	Required
Microsoft.VisualStudio.Component.IntelliCode	IntelliCode	0.1	Required
Microsoft.VisualStudio.Component.Merq	Common Xamarin internal tools	16.2.29012.281	Required
Microsoft.VisualStudio.Component.MonoDebugger	Mono debugger	16.0.28517.75	Required
Microsoft.VisualStudio.Component.NuGet	NuGet package manager	16.1.28829.92	Required
Microsoft.VisualStudio.Component.Roslyn.Compiler	C# and Visual Basic Roslyn compilers	16.0.28714.129	Required

Component ID	Name	Version	Dependency Type
Microsoft.VisualStudio.Component.Roslyn.LanguageServices	C# and Visual Basic	16.4.29429.68	Required
Microsoft.VisualStudio.ComponentGroup.WebToolsExtensions.TemplateEngine	ASP.NET templating engine	16.0.28315.86	Required
Component.Android.SDK28	Android SDK setup (API level 28)	16.2.29003.222	Recommended

ASP.NET and web development

ID: Microsoft.VisualStudio.Workload.NetWeb

Description: Build web applications using ASP.NET Core, ASP.NET, HTML/JavaScript, and Containers including Docker support.

Components included by this workload

Component ID	Name	Version	Dependency Type
Component.Microsoft.VisualStudio.RazorExtension	Razor Language Services	16.0.28714.129	Required
Component.Microsoft.Web.LibraryManager	Library Manager	16.0.28315.86	Required
Microsoft.Component.MSBuild	MSBuild	16.4.29429.68	Required
Microsoft.Net.Component.4.5.2.TargetingPack	.NET Framework 4.5.2 targeting pack	16.0.28517.75	Required
Microsoft.Net.Component.4.5.TargetingPack	.NET Framework 4.5 targeting pack	16.0.28517.75	Required
Microsoft.Net.Component.4.6.1.TargetingPack	.NET Framework 4.6.1 targeting pack	16.0.28517.75	Required
Microsoft.Net.Component.4.7.2.TargetingPack	.NET Framework 4.7.2 targeting pack	16.0.28517.75	Required
Microsoft.Net.Component.4.8.SDK	.NET Framework 4.8 SDK	16.4.29313.120	Required
Microsoft.Net.ComponentGroup.DevelopmentPrerequisites	.NET Framework 4.7.2 development tools	16.3.29207.166	Required
Microsoft.NetCore.Component.DevelopmentTools	.NET Core development tools	16.4.29511.114	Required

Component ID	Name	Version	Dependency Type
Microsoft.NetCore.Component.SDK	.NET Core 3.1 SDK	16.4.29519.181	Required
Microsoft.NetCore.Component.Web	.NET Core development tools	16.4.29511.114	Required
Microsoft.VisualStudio.Component.Common.Azure.Tools	Connectivity and publishing tools	16.4.29409.204	Required
Microsoft.VisualStudio.Component.DockerTools	Container development tools	16.4.29409.204	Required
Microsoft.VisualStudio.Component.FSharp	F# language support	16.0.28315.86	Required
Microsoft.VisualStudio.Component.FSharp.WebTemplates	F# language support for web projects	16.3.29207.166	Required
Microsoft.VisualStudio.Component.IISExpress	IIS Express	16.0.28315.86	Required
Microsoft.VisualStudio.Component.IntelliCode	IntelliCode	0.1	Required
Microsoft.VisualStudio.Component.JavaScript.Diagnostics	JavaScript diagnostics	16.0.28517.75	Required
Microsoft.VisualStudio.Component.JavaScript.TypeScript	JavaScript and TypeScript language support	16.4.29409.204	Required
Microsoft.VisualStudio.Component.ManagedDesktop.Core	Managed Desktop Workload Core	16.4.29318.151	Required
Microsoft.VisualStudio.Component.MSODBC.SQL	SQL Server ODBC Driver	16.0.28625.61	Required
Microsoft.VisualStudio.Component.MSSQL.CMDLnUtils	SQL Server Command Line Utilities	16.0.28707.177	Required
Microsoft.VisualStudio.Component.NuGet	NuGet package manager	16.1.28829.92	Required
Microsoft.VisualStudio.Component.Roslyn.Compiler	C# and Visual Basic Roslyn compilers	16.0.28714.129	Required
Microsoft.VisualStudio.Component.Roslyn.LanguageServices	C# and Visual Basic	16.4.29429.68	Required
Microsoft.VisualStudio.Component.SQL.ADAL	SQL ADAL runtime	16.0.28517.75	Required

Component ID	Name	Version	Dependency Type
Microsoft.VisualStudio.Component.SQL.CLR	CLR data types for SQL Server	16.0.28315.86	Required
Microsoft.VisualStudio.Component.SQL.DataSources	Data sources for SQL Server support	16.0.28315.86	Required
Microsoft.VisualStudio.Component.SQL.LocalDB.Runtime	SQL Server Express 2016 LocalDB	16.0.28625.61	Required
Microsoft.VisualStudio.Component.SQL.SSDT	SQL Server Data Tools	16.3.29207.166	Required
Microsoft.VisualStudio.Component.TextTemplating	Text Template Transformation	16.0.28625.61	Required
Microsoft.VisualStudio.Component.TypeScript.3.7	TypeScript 3.7 SDK	16.0.29429.68	Required
Microsoft.VisualStudio.Component.Web	ASP.NET and web development tools	16.0.28517.75	Required
Microsoft.VisualStudio.ComponentGroup.Web	ASP.NET and web development tools prerequisites	16.4.29318.151	Required
Microsoft.VisualStudio.ComponentGroup.WebToolsExtensions	ASP.NET and web development	16.0.28621.142	Required
Component.Microsoft.VisualStudio.LiveShare	Live Share	1.0.1076	Recommended
Component.Microsoft.VisualStudio.Web.AzureFunctions	Azure WebJobs Tools	16.0.28714.129	Recommended
Microsoft.Net.Component.4.5.1.TargetingPack	.NET Framework 4.5.1 targeting pack	16.0.28517.75	Recommended
Microsoft.Net.Component.4.6.TargetingPack	.NET Framework 4.6 targeting pack	16.0.28517.75	Recommended
Microsoft.Net.Component.4.TargetingPack	.NET Framework 4 targeting pack	16.0.28517.75	Recommended
Microsoft.Net.ComponentGroup.TargetingPacks.Common	.NET Framework 4 – 4.6 development tools	16.0.28516.191	Recommended
Microsoft.Net.Core.Component.SDK.2.1	.NET Core 2.1 LTS Runtime	16.4.29519.181	Recommended
Microsoft.VisualStudio.Component.AppInsights.Tools	Developer Analytics tools	16.4.29429.68	Recommended

Component ID	Name	Version	Dependency Type
Microsoft.VisualStudio.Component.AspNet45	Advanced ASP.NET features	16.0.28315.86	Recommended
Microsoft.VisualStudio.Component.Azure.AuthoringTools	Azure Authoring Tools	16.0.28625.61	Recommended
Microsoft.VisualStudio.Component.Azure.ClientLibs	Azure libraries for .NET	16.0.28315.86	Recommended
Microsoft.VisualStudio.Component.Azure.Compute.Emulator	Azure Compute Emulator	16.1.28810.153	Recommended
Microsoft.VisualStudio.Component.Azure.Storage.Emulator	Azure Storage Emulator	16.4.29313.120	Recommended
Microsoft.VisualStudio.Component.CloudExplorer	Cloud Explorer	16.0.28625.61	Recommended
Microsoft.VisualStudio.Component.Debugger.Snapshot	Snapshot Debugger	16.2.29003.222	Recommended
Microsoft.VisualStudio.Component.Debugger.TimeTravel	Time Travel Debugger	16.4.29511.114	Recommended
Microsoft.VisualStudio.Component.DiagnosticTools	.NET profiling tools	16.4.29429.68	Recommended
Microsoft.VisualStudio.Component.EntityFramework	Entity Framework 6 tools	16.0.28315.86	Recommended
Microsoft.VisualStudio.Component.IntelliTrace.FrontEnd	IntelliTrace	16.4.29429.68	Recommended
Microsoft.VisualStudio.Component.LiveUnitTesting	Live Unit Testing	16.1.28811.260	Recommended
Microsoft.VisualStudio.Component.WebDeploy	Web Deploy	16.0.28517.75	Recommended
Microsoft.VisualStudio.ComponentGroup.AzureFunctions	Azure WebJobs Tools	16.0.28621.142	Recommended
Microsoft.VisualStudio.ComponentGroup.Web.CloudTools	Cloud tools for web development	16.2.29003.222	Recommended
Microsoft.Net.Component.4.6.2.TargetingPack	.NET Framework 4.6.2 targeting pack	16.0.28517.75	Optional
Microsoft.Net.Component.4.7.1.TargetingPack	.NET Framework 4.7.1 targeting pack	16.0.28517.75	Optional

Component ID	Name	Version	Dependency Type
Microsoft.Net.Component.4.7.TargetingPack	.NET Framework 4.7 targeting pack	16.0.28517.75	Optional
Microsoft.Net.Component.4.8.TargetingPack	.NET Framework 4.8 targeting pack	16.4.29313.120	Optional
Microsoft.Net.ComponentGroup.4.6.1.DeveloperTools	.NET Framework 4.6.1 development tools	16.3.29207.166	Optional
Microsoft.Net.ComponentGroup.4.6.2.DeveloperTools	.NET Framework 4.6.2 development tools	16.3.29207.166	Optional
Microsoft.Net.ComponentGroup.4.7.1.DeveloperTools	.NET Framework 4.7.1 development tools	16.3.29207.166	Optional
Microsoft.Net.ComponentGroup.4.7.DeveloperTools	.NET Framework 4.7 development tools	16.3.29207.166	Optional
Microsoft.Net.ComponentGroup.4.8.DeveloperTools	.NET Framework 4.8 development tools	16.4.29318.151	Optional
Microsoft.VisualStudio.Component.ClassDesigner	Class Designer	16.0.28528.71	Optional
Microsoft.VisualStudio.Component.CodeClone	Code Clone	16.4.29409.204	Optional
Microsoft.VisualStudio.Component.CodeMap	Code Map	16.0.28625.61	Optional
Microsoft.VisualStudio.Component.DependencyValidation.Enterprise	Live Dependency Validation	16.0.28625.61	Optional
Microsoft.VisualStudio.Component.GraphDocument	DGML editor	16.0.28625.61	Optional
Microsoft.VisualStudio.Component.TestTools.WebLoadTest	Web performance and load testing tools	16.0.28625.61	Optional
Microsoft.VisualStudio.Component.Wcf.Tooling	Windows Communication Foundation	16.0.28625.61	Optional
Microsoft.VisualStudio.ComponentGroup.AdditionalWebProjectTemplates	Additional project templates (previous versions)	16.0.28621.142	Optional
Microsoft.VisualStudio.ComponentGroup.ArchitectureTools.Managed	Architecture and analysis tools	16.4.29429.68	Optional
Microsoft.VisualStudio.ComponentGroup.IISDevelopment	Development time IIS support	16.0.28315.86	Optional

Node.js development

ID: Microsoft.VisualStudio.Workload.Node

Description: Build scalable network applications using Node.js, an asynchronous event-driven JavaScript runtime.

Components included by this workload

Component ID	Name	Version	Dependency Type
Microsoft.VisualStudio.Component.JavaScript.Diagnostics	JavaScript diagnostics	16.0.28517.75	Required
Microsoft.VisualStudio.Component.JavaScript.TypeScript	JavaScript and TypeScript language support	16.4.29409.204	Required
Microsoft.VisualStudio.Component.Node.Tools	Node.js development tools	16.4.29429.68	Required
Microsoft.VisualStudio.Component.TypeScript.3.7	TypeScript 3.7 SDK	16.0.29429.68	Required
Microsoft.VisualStudio.ComponentGroup.WebToolsExtensions	ASP.NET and web development	16.0.28621.142	Required
Component.Microsoft.VisualStudio.LiveShare	Live Share	1.0.1076	Recommended
Microsoft.VisualStudio.Component.IntelliCode	IntelliCode	0.1	Recommended
Microsoft.VisualStudio.Component.WebDeploy	Web Deploy	16.0.28517.75	Recommended
Microsoft.VisualStudio.Component.AppInsights.Tools	Developer Analytics tools	16.4.29429.68	Optional
Microsoft.VisualStudio.Component.Common.Azure.Tools	Connectivity and publishing tools	16.4.29409.204	Optional
Microsoft.VisualStudio.Component.VC.Corelde	C++ core features	16.0.28625.61	Optional
Microsoft.VisualStudio.Component.VC.Tools.x86.x64	MSVC v142 - VS 2019 C++ x64/x86 build tools (v14.24)	16.4.29409.204	Optional

Office/SharePoint development

ID: Microsoft.VisualStudio.Workload.Office

Description: Create Office and SharePoint add-ins, SharePoint solutions, and VSTO add-ins using C#, VB, and JavaScript.

Components included by this workload

Component ID	Name	Version	Dependency Type
Component.Microsoft.VisualStudio.RazorExtension	Razor Language Services	16.0.28714.129	Required
Component.Microsoft.Web.LibraryManager	Library Manager	16.0.28315.86	Required
Microsoft.Component.MSBuild	MSBuild	16.4.29429.68	Required
Microsoft.Net.Component.4.5.2.TargetingPack	.NET Framework 4.5.2 targeting pack	16.0.28517.75	Required
Microsoft.Net.Component.4.5.TargetingPack	.NET Framework 4.5 targeting pack	16.0.28517.75	Required
Microsoft.Net.Component.4.6.1.TargetingPack	.NET Framework 4.6.1 targeting pack	16.0.28517.75	Required
Microsoft.Net.Component.4.7.2.TargetingPack	.NET Framework 4.7.2 targeting pack	16.0.28517.75	Required
Microsoft.Net.Component.4.8.SDK	.NET Framework 4.8 SDK	16.4.29313.120	Required
Microsoft.Net.Component.4.TargetingPack	.NET Framework 4 targeting pack	16.0.28517.75	Required
Microsoft.Net.ComponentGroup.DevelopmentPrerequisites	.NET Framework 4.7.2 development tools	16.3.29207.166	Required
Microsoft.NetCore.Component.SDK	.NET Core 3.1 SDK	16.4.29519.181	Required
Microsoft.VisualStudio.Component.AppInsights.Tools	Developer Analytics tools	16.4.29429.68	Required
Microsoft.VisualStudio.Component.Common.Azure.Tools	Connectivity and publishing tools	16.4.29409.204	Required
Microsoft.VisualStudio.Component.DockerTools	Container development tools	16.4.29409.204	Required
Microsoft.VisualStudio.Component.IISExpress	IIS Express	16.0.28315.86	Required
Microsoft.VisualStudio.Component.IntelliCode	IntelliCode	0.1	Required
Microsoft.VisualStudio.Component.JavaScript.Diagnostics	JavaScript diagnostics	16.0.28517.75	Required
Microsoft.VisualStudio.Component.JavaScript.TypeScript	JavaScript and TypeScript language support	16.4.29409.204	Required

Component ID	Name	Version	Dependency Type
Microsoft.VisualStudio.Component.ManagedDesktop.Core	Managed Desktop Workload Core	16.4.29318.151	Required
Microsoft.VisualStudio.Component.ManagedDesktop.Prerequisites	.NET desktop development tools	16.4.29429.68	Required
Microsoft.VisualStudio.Component.MSODBC.SQL	SQL Server ODBC Driver	16.0.28625.61	Required
Microsoft.VisualStudio.Component.MSSQL.CMDLnUtils	SQL Server Command Line Utilities	16.0.28707.177	Required
Microsoft.VisualStudio.Component.NuGet	NuGet package manager	16.1.28829.92	Required
Microsoft.VisualStudio.Component.Roslyn.Compiler	C# and Visual Basic Roslyn compilers	16.0.28714.129	Required
Microsoft.VisualStudio.Component.Roslyn.LanguageServices	C# and Visual Basic	16.4.29429.68	Required
Microsoft.VisualStudio.Component.Sharepoint.Tools	Office Developer Tools for Visual Studio	16.4.29409.204	Required
Microsoft.VisualStudio.Component.SQL.ADAL	SQL ADAL runtime	16.0.28517.75	Required
Microsoft.VisualStudio.Component.SQL.CLR	CLR data types for SQL Server	16.0.28315.86	Required
Microsoft.VisualStudio.Component.SQL.DataSources	Data sources for SQL Server support	16.0.28315.86	Required
Microsoft.VisualStudio.Component.SQL.LocalDB.Runtime	SQL Server Express 2016 LocalDB	16.0.28625.61	Required
Microsoft.VisualStudio.Component.SQL.SSDT	SQL Server Data Tools	16.3.29207.166	Required
Microsoft.VisualStudio.Component.TextTemplating	Text Template Transformation	16.0.28625.61	Required
Microsoft.VisualStudio.Component.TypeScript.3.7	TypeScript 3.7 SDK	16.0.29429.68	Required
Microsoft.VisualStudio.Component.Wcf.Tooling	Windows Communication Foundation	16.0.28625.61	Required
Microsoft.VisualStudio.Component.Web	ASP.NET and web development tools	16.0.28517.75	Required

Component ID	Name	Version	Dependency Type
Microsoft.VisualStudio.Component.Workflow	Windows Workflow Foundation	16.0.28315.86	Required
Microsoft.VisualStudio.ComponentGroup.Web	ASP.NET and web development tools prerequisites	16.4.29318.151	Required
Microsoft.VisualStudio.ComponentGroup.WebToolsExtensions	ASP.NET and web development	16.0.28621.142	Required
Microsoft.VisualStudio.Component.TeamOffice	Visual Studio Tools for Office (VSTO)	16.4.29409.204	Recommended
Microsoft.VisualStudio.Component.WebDeploy	Web Deploy	16.0.28517.75	Recommended
Microsoft.Net.Component.4.6.2.TargetingPack	.NET Framework 4.6.2 targeting pack	16.0.28517.75	Optional
Microsoft.Net.Component.4.7.1.TargetingPack	.NET Framework 4.7.1 targeting pack	16.0.28517.75	Optional
Microsoft.Net.Component.4.7.TargetingPack	.NET Framework 4.7 targeting pack	16.0.28517.75	Optional
Microsoft.Net.Component.4.8.TargetingPack	.NET Framework 4.8 targeting pack	16.4.29313.120	Optional
Microsoft.Net.ComponentGroup.4.6.1.DeveloperTools	.NET Framework 4.6.1 development tools	16.3.29207.166	Optional
Microsoft.Net.ComponentGroup.4.6.2.DeveloperTools	.NET Framework 4.6.2 development tools	16.3.29207.166	Optional
Microsoft.Net.ComponentGroup.4.7.1.DeveloperTools	.NET Framework 4.7.1 development tools	16.3.29207.166	Optional
Microsoft.Net.ComponentGroup.4.7.DeveloperTools	.NET Framework 4.7 development tools	16.3.29207.166	Optional
Microsoft.Net.ComponentGroup.4.8.DeveloperTools	.NET Framework 4.8 development tools	16.4.29318.151	Optional
Microsoft.VisualStudio.Component.IntelliTrace.FrontEnd	IntelliTrace	16.4.29429.68	Optional
Microsoft.VisualStudio.ComponentGroup.Sharepoint.WIF	Windows Identity Foundation 3.5	16.0.28621.142	Optional

Python development

ID: Microsoft.VisualStudio.Workload.Python

Description: Editing, debugging, interactive development and source control for Python.

Components included by this workload

Component ID	Name	Version	Dependency Type
Microsoft.Component.PythonTools	Python language support	16.4.29429.68	Required
Component.CPython3.x64	Python 3 64-bit (3.7.5)	3.7.5	Recommended
Component.Microsoft.VisualStudio.LiveShare	Live Share	1.0.1076	Recommended
Microsoft.Component.PythonTools.Minicondax64	Python miniconda	16.2.29003.222	Recommended
Microsoft.Component.PythonTools.Web	Python web support	16.0.28517.75	Recommended
Microsoft.VisualStudio.Component.Common.Azure.Tools	Connectivity and publishing tools	16.4.29409.204	Recommended
Microsoft.VisualStudio.Component.JavaScript.TypeScript	JavaScript and TypeScript language support	16.4.29409.204	Recommended
Microsoft.VisualStudio.Component.TypeScript.3.7	TypeScript 3.7 SDK	16.0.29429.68	Recommended
Microsoft.VisualStudio.Component.WebDeploy	Web Deploy	16.0.28517.75	Recommended
Microsoft.VisualStudio.ComponentGroup.WebToolsExtensions	ASP.NET and web development	16.0.28621.142	Recommended
Component.CPython2.x64	Python 2 64-bit (2.7.16)	2.7.16	Optional
Component.CPython2.x86	Python 2 32-bit (2.7.16)	2.7.16	Optional
Component.CPython3.x86	Python 3 32-bit (3.7.5)	3.7.5	Optional
Component.Microsoft.VisualStudio.RazorExtension	Razor Language Services	16.0.28714.129	Optional
Component.Microsoft.Web.LibraryManager	Library Manager	16.0.28315.86	Optional
Microsoft.Component.MSBuild	MSBuild	16.4.29429.68	Optional
Microsoft.ComponentGroup.PythonTools.NativeDevelopment	Python native development tools	16.2.29020.229	Optional

Component ID	Name	Version	Dependency Type
Microsoft.Net.Component.4.5.2.TargetingPack	.NET Framework 4.5.2 targeting pack	16.0.28517.75	Optional
Microsoft.Net.Component.4.5.TargetingPack	.NET Framework 4.5 targeting pack	16.0.28517.75	Optional
Microsoft.Net.Component.4.7.2.TargetingPack	.NET Framework 4.7.2 targeting pack	16.0.28517.75	Optional
Microsoft.Net.Component.4.8.SDK	.NET Framework 4.8 SDK	16.4.29313.120	Optional
Microsoft.Net.ComponentGroup.DevelopmentPrerequisites	.NET Framework 4.7.2 development tools	16.3.29207.166	Optional
Microsoft.NetCore.Component.SDK	.NET Core 3.1 SDK	16.4.29519.181	Optional
Microsoft.VisualStudio.Component.Azure.AuthoringTools	Azure Authoring Tools	16.0.28625.61	Optional
Microsoft.VisualStudio.Component.Azure.ClientLibs	Azure libraries for .NET	16.0.28315.86	Optional
Microsoft.VisualStudio.Component.Azure.Compute.Emulator	Azure Compute Emulator	16.1.28810.153	Optional
Microsoft.VisualStudio.Component.Azure.Storage.Emulator	Azure Storage Emulator	16.4.29313.120	Optional
Microsoft.VisualStudio.Component.Azure.Waverton	Azure Cloud Services core tools	16.4.29409.204	Optional
Microsoft.VisualStudio.Component.Azure.Waverton.BuildTools	Azure Cloud Services build tools	16.3.29207.166	Optional
Microsoft.VisualStudio.Component.DockerTools	Container development tools	16.4.29409.204	Optional
Microsoft.VisualStudio.Component.Graphics.Tools	Graphics debugger and GPU profiler for DirectX	16.0.28625.61	Optional
Microsoft.VisualStudio.Component.IISExpress	IIS Express	16.0.28315.86	Optional
Microsoft.VisualStudio.Component.JavaScript.Diagnostics	JavaScript diagnostics	16.0.28517.75	Optional

Component ID	Name	Version	Dependency Type
Microsoft.VisualStudio.Component.ManagedDesktop.Core	Managed Desktop Workload Core	16.4.29318.151	Optional
Microsoft.VisualStudio.Component.MSODBC.SQL	SQL Server ODBC Driver	16.0.28625.61	Optional
Microsoft.VisualStudio.Component.MSSQL.CMDLnUtils	SQL Server Command Line Utilities	16.0.28707.177	Optional
Microsoft.VisualStudio.Component.NuGet	NuGet package manager	16.1.28829.92	Optional
Microsoft.VisualStudio.Component.Roslyn.Compiler	C# and Visual Basic Roslyn compilers	16.0.28714.129	Optional
Microsoft.VisualStudio.Component.Roslyn.LanguageServices	C# and Visual Basic	16.4.29429.68	Optional
Microsoft.VisualStudio.Component.SQL.ADAL	SQL ADAL runtime	16.0.28517.75	Optional
Microsoft.VisualStudio.Component.SQL.CLR	CLR data types for SQL Server	16.0.28315.86	Optional
Microsoft.VisualStudio.Component.SQL.DataSources	Data sources for SQL Server support	16.0.28315.86	Optional
Microsoft.VisualStudio.Component.SQL.LocalDB.Runtime	SQL Server Express 2016 LocalDB	16.0.28625.61	Optional
Microsoft.VisualStudio.Component.SQL.SSDT	SQL Server Data Tools	16.3.29207.166	Optional
Microsoft.VisualStudio.Component.TextTemplating	Text Template Transformation	16.0.28625.61	Optional
Microsoft.VisualStudio.Component.VC.CoreLDE	C++ core features	16.0.28625.61	Optional
Microsoft.VisualStudio.Component.VC.DiagnosticTools	C++ profiling tools	16.4.29429.68	Optional
Microsoft.VisualStudio.Component.VC.Tools.x86.x64	MSVC v142 - VS 2019 C++ x64/x86 build tools (v14.24)	16.4.29409.204	Optional
Microsoft.VisualStudio.Component.Web	ASP.NET and web development tools	16.0.28517.75	Optional
Microsoft.VisualStudio.Component.Windows10SDK	Windows Universal C Runtime	16.4.29409.204	Optional

Component ID	Name	Version	Dependency Type
Microsoft.VisualStudio.Component.Windows10SDK.18362	Windows 10 SDK (10.0.18362.0)	16.1.28829.92	Optional
Microsoft.VisualStudio.ComponentGroup.Web	ASP.NET and web development tools prerequisites	16.4.29318.151	Optional

Universal Windows Platform development

ID: Microsoft.VisualStudio.Workload.Universal

Description: Create applications for the Universal Windows Platform with C#, VB, or optionally C++.

Components included by this workload

Component ID	Name	Version	Dependency Type
Microsoft.Component.NetFX.Native	.NET Native	16.4.29429.68	Required
Microsoft.ComponentGroup.Blend	Blend for Visual Studio	16.0.28315.86	Required
Microsoft.Net.Component.4.5.TargetingPack	.NET Framework 4.5 targeting pack	16.0.28517.75	Required
Microsoft.NetCore.Component.SDK	.NET Core 3.1 SDK	16.4.29519.181	Required
Microsoft.VisualStudio.Component.AppInsights.Tools	Developer Analytics tools	16.4.29429.68	Required
Microsoft.VisualStudio.Component.DiagnosticTools	.NET profiling tools	16.4.29429.68	Required
Microsoft.VisualStudio.Component.Graphics	Image and 3D model editors	16.0.28517.75	Required
Microsoft.VisualStudio.Component.IntelliCode	IntelliCode	0.1	Required
Microsoft.VisualStudio.Component.NuGet	NuGet package manager	16.1.28829.92	Required
Microsoft.VisualStudio.Component.Roslyn.Compiler	C# and Visual Basic Roslyn compilers	16.0.28714.129	Required
Microsoft.VisualStudio.Component.Roslyn.LanguageServices	C# and Visual Basic	16.4.29429.68	Required
Microsoft.VisualStudio.Component.SQL CLR	CLR data types for SQL Server	16.0.28315.86	Required

Component ID	Name	Version	Dependency Type
Microsoft.VisualStudio.Component.Windows10SDK.18362	Windows 10 SDK (10.0.18362.0)	16.1.28829.92	Required
Microsoft.VisualStudio.ComponentGroup.MSIX.Packaging	MSIX Packaging Tools	16.4.29409.204	Required
Microsoft.VisualStudio.ComponentGroup.UWP.NetCoreAndStandard	.NET Native and .NET Standard	16.3.29102.218	Required
Microsoft.VisualStudio.ComponentGroup.UWP.Support	Universal Windows Platform tools	16.4.29409.204	Required
Microsoft.VisualStudio.ComponentGroup.UWP.Xamarin	Universal Windows Platform tools for Xamarin	16.4.29511.114	Required
Microsoft.VisualStudio.Component.IntelliTrace.FrontEnd	IntelliTrace	16.4.29429.68	Recommended
Microsoft.Component.MSBuild	MSBuild	16.4.29429.68	Optional
Microsoft.Net.Component.4.8.SDK	.NET Framework 4.8 SDK	16.4.29313.120	Optional
Microsoft.VisualStudio.Component.ClassDesigner	Class Designer	16.0.28528.71	Optional
Microsoft.VisualStudio.Component.CodeClone	Code Clone	16.4.29409.204	Optional
Microsoft.VisualStudio.Component.CodeMap	Code Map	16.0.28625.61	Optional
Microsoft.VisualStudio.Component.DependencyValidation.Enterprise	Live Dependency Validation	16.0.28625.61	Optional
Microsoft.VisualStudio.Component.GraphDocument	DGML editor	16.0.28625.61	Optional
Microsoft.VisualStudio.Component.Graphics.Tools	Graphics debugger and GPU profiler for DirectX	16.0.28625.61	Optional
Microsoft.VisualStudio.Component.SQL.LocalDB.Runtime	SQL Server Express 2016 LocalDB	16.0.28625.61	Optional
Microsoft.VisualStudio.Component.TextTemplating	Text Template Transformation	16.0.28625.61	Optional
Microsoft.VisualStudio.Component.UWP.VC.ARM64	C++ Universal Windows Platform support for v142 build tools (ARM64)	16.3.29207.166	Optional

Component ID	Name	Version	Dependency Type
Microsoft.VisualStudio.Component.VC.CoreRedistributable	C++ core features	16.0.28625.61	Optional
Microsoft.VisualStudio.Component.VC.Redist.14.Latest	C++ 2019 Redistributable Update	16.4.29429.68	Optional
Microsoft.VisualStudio.Component.VC.Tools.ARM	MSVC v142 - VS 2019 C++ ARM build tools (v14.24)	16.4.29409.204	Optional
Microsoft.VisualStudio.Component.VC.Tools.ARM64	MSVC v142 - VS 2019 C++ ARM64 build tools (v14.24)	16.4.29409.204	Optional
Microsoft.VisualStudio.Component.VC.Tools.x86.x64	MSVC v142 - VS 2019 C++ x64/x86 build tools (v14.24)	16.4.29409.204	Optional
Microsoft.VisualStudio.Component.VC.v141.ARM	MSVC v141 - VS 2017 C++ ARM build tools (v14.16)	16.2.29003.222	Optional
Microsoft.VisualStudio.Component.VC.v141.ARM64	MSVC v141 - VS 2017 C++ ARM64 build tools (v14.16)	16.1.28829.92	Optional
Microsoft.VisualStudio.Component.VC.v141.x86.x64	MSVC v141 - VS 2017 C++ x64/x86 build tools (v14.16)	16.1.28829.92	Optional
Microsoft.VisualStudio.Component.Windows10SDK.1629.9	Windows 10 SDK (10.0.16299.0)	16.0.28517.75	Optional
Microsoft.VisualStudio.Component.Windows10SDK.1713.4	Windows 10 SDK (10.0.17134.0)	16.0.28517.75	Optional
Microsoft.VisualStudio.Component.Windows10SDK.1776.3	Windows 10 SDK (10.0.17763.0)	16.0.28517.75	Optional
Microsoft.VisualStudio.Component.Windows10SDK.IoOverUsb	USB Device Connectivity	16.4.29511.114	Optional
Microsoft.VisualStudio.ComponentGroup.ArchitectureTools.Managed	Architecture and analysis tools	16.4.29429.68	Optional
Microsoft.VisualStudio.ComponentGroup.ArchitectureTools.Native	Architecture tools for C++	16.0.28621.142	Optional
Microsoft.VisualStudio.ComponentGroup.NativeDesktop.Core	C++ core desktop features	16.2.29012.281	Optional
Microsoft.VisualStudio.ComponentGroup.UWP.VC	C++ (v142) Universal Windows Platform tools	16.3.29207.166	Optional

Component ID	Name	Version	Dependency Type
Microsoft.VisualStudio.ComponentGroup.UWP.VC.v141	C++ (v141) Universal Windows Platform tools	16.1.28810.153	Optional

Visual Studio extension development

ID: Microsoft.VisualStudio.Workload.VisualStudioExtension

Description: Create add-ons and extensions for Visual Studio, including new commands, code analyzers and tool windows.

Components included by this workload

Component ID	Name	Version	Dependency Type
Microsoft.Component.MSBuild	MSBuild	16.4.29429.68	Required
Microsoft.Net.Component.4.6.TargetingPack	.NET Framework 4.6 targeting pack	16.0.28517.75	Required
Microsoft.Net.Component.4.7.2.TargetingPack	.NET Framework 4.7.2 targeting pack	16.0.28517.75	Required
Microsoft.Net.Component.4.8.SDK	.NET Framework 4.8 SDK	16.4.29313.120	Required
Microsoft.Net.ComponentGroup.DevelopmentPrerequisites	.NET Framework 4.7.2 development tools	16.3.29207.166	Required
Microsoft.VisualStudio.Component.IntelliCode	IntelliCode	0.1	Required
Microsoft.VisualStudio.Component.NuGet	NuGet package manager	16.1.28829.92	Required
Microsoft.VisualStudio.Component.Roslyn.Compiler	C# and Visual Basic Roslyn compilers	16.0.28714.129	Required
Microsoft.VisualStudio.Component.Roslyn.LanguageServices	C# and Visual Basic	16.4.29429.68	Required
Microsoft.VisualStudio.Component.VSSDK	Visual Studio SDK	16.0.28315.86	Required
Microsoft.VisualStudio.ComponentGroup.VisualStudioExtension.Prerequisites	Visual Studio extension development prerequisites	16.4.29318.151	Required
Microsoft.VisualStudio.Component.DiagnosticTools	.NET profiling tools	16.4.29429.68	Recommended

Component ID	Name	Version	Dependency Type
Microsoft.VisualStudio.Component.IntelliTrace.FrontEnd	IntelliTrace	16.4.29429.68	Recommended
Microsoft.VisualStudio.Component.TextTemplating	Text Template Transformation	16.0.28625.61	Recommended
Microsoft.Component.CodeAnalysis.SDK	.NET Compiler Platform SDK	16.2.29003.222	Optional
Microsoft.VisualStudio.Component.AppInsights.Tools	Developer Analytics tools	16.4.29429.68	Optional
Microsoft.VisualStudio.Component.DslTools	Modeling SDK	16.0.28315.86	Optional

Unaffiliated components

These are components that are not included with any workload, but may be selected as an individual component.

Component ID	Name	Version
Component.GitHub.VisualStudio	GitHub extension for Visual Studio	2.5.9.5485
Component.Xamarin.Inspector	Xamarin Inspector	16.0.28315.86
Component.Xamarin.Profiler	Xamarin Profiler	16.0.28315.86
Component.Xamarin.Workbooks	Xamarin Workbooks	16.0.28315.86
Microsoft.Component.ClickOnce	ClickOnce Publishing	16.4.29409.204
Microsoft.Component.HelpViewer	Help Viewer	16.0.28625.61
Microsoft.Net.Component.4.6.1.SDK	.NET Framework 4.6.1 SDK	16.4.29409.204
Microsoft.Net.Component.4.6.2.SDK	.NET Framework 4.6.2 SDK	16.4.29409.204
Microsoft.Net.Component.4.7.1.SDK	.NET Framework 4.7.1 SDK	16.4.29409.204
Microsoft.Net.Component.4.7.2.SDK	.NET Framework 4.7.2 SDK	16.4.29409.204
Microsoft.Net.Component.4.7.SDK	.NET Framework 4.7 SDK	16.4.29409.204
Microsoft.Net.Core.Component.SDK.2.2	.NET Core 2.2 Runtime	16.4.29519.181
Microsoft.Net.Core.Component.SDK.3.0	.NET Core 3.0 Runtime	16.4.29519.181
Microsoft.NetCore.ComponentGroup.DevelopmentTools.2.1	Development Tools plus .NET Core 2.1	16.3.29207.166

Component ID	Name	Version
Microsoft.NetCore.ComponentGroup.Web.2.1	Web Development Tools plus .NET Core 2.1	16.3.29207.166
Microsoft.VisualStudio.Component.AzureDevOps.OfficeIntegration	Azure DevOps Office Integration	16.0.28625.61
Microsoft.VisualStudio.Component.Debugger.VSOnline	Debugger for Visual Studio Online Environments	16.4.29429.68
Microsoft.VisualStudio.Component.Git	Git for Windows	16.0.28625.61
Microsoft.VisualStudio.Component.LinqToSql	LINQ to SQL tools	16.0.28625.61
Microsoft.VisualStudio.Component.TestTools.CodedUITest	Coded UI test	16.0.28327.66
Microsoft.VisualStudio.Component.VC.14.20.ARM	MSVC v142 - VS 2019 C++ ARM build tools (v14.20)	16.4.29511.114
Microsoft.VisualStudio.Component.VC.14.20.ARM.Spectre	MSVC v142 - VS 2019 C++ ARM Spectre-mitigated libs (v14.20)	16.4.29511.114
Microsoft.VisualStudio.Component.VC.14.20.ARM64	MSVC v142 - VS 2019 C++ ARM64 build tools (v14.20)	16.4.29511.114
Microsoft.VisualStudio.Component.VC.14.20.ARM64.Spectre	MSVC v142 - VS 2019 C++ ARM64 Spectre-mitigated libs (v14.20)	16.4.29511.114
Microsoft.VisualStudio.Component.VC.14.20.ATL	C++ v14.20 ATL for v142 build tools (x86 & x64)	16.1.28829.92
Microsoft.VisualStudio.Component.VC.14.20.ATL.ARM	C++ v14.20 ATL for v142 build tools (ARM)	16.4.29511.114
Microsoft.VisualStudio.Component.VC.14.20.ATL.ARM.Spectre	C++ v14.20 ATL for v142 build tools with Spectre Mitigations (ARM)	16.4.29511.114
Microsoft.VisualStudio.Component.VC.14.20.ATL.ARM64	C++ v14.20 ATL for v142 build tools (ARM64)	16.4.29511.114
Microsoft.VisualStudio.Component.VC.14.20.ATL.ARM64.Spectre	C++ v14.20 ATL for v142 build tools with Spectre Mitigations (ARM64)	16.4.29511.114
Microsoft.VisualStudio.Component.VC.14.20.ATL.Spectre	C++ v14.20 ATL for v142 build tools with Spectre Mitigations (x86 & x64)	16.4.29511.114
Microsoft.VisualStudio.Component.VC.14.20.CLI.Support	C++/CLI support for v142 build tools (14.20)	16.4.29409.204
Microsoft.VisualStudio.Component.VC.14.20.MFC	C++ v14.20 MFC for v142 build tools (x86 & x64)	16.2.29003.222

Component ID	Name	Version
Microsoft.VisualStudio.Component.VC.14.20.MFC.ARM	C++ v14.20 MFC for v142 build tools (ARM)	16.4.29511.114
Microsoft.VisualStudio.Component.VC.14.20.MFC.ARM.Spectre	C++ v14.20 MFC for v142 build tools with Spectre Mitigations (ARM)	16.4.29511.114
Microsoft.VisualStudio.Component.VC.14.20.MFC.ARM64	C++ v14.20 MFC for v142 build tools (ARM64)	16.4.29511.114
Microsoft.VisualStudio.Component.VC.14.20.MFC.ARM64.Spectre	C++ v14.20 MFC for v142 build tools with Spectre Mitigations (ARM64)	16.4.29511.114
Microsoft.VisualStudio.Component.VC.14.20.MFC.Spectre	C++ v14.20 MFC for v142 build tools with Spectre Mitigations (x86 & x64)	16.4.29511.114
Microsoft.VisualStudio.Component.VC.14.20.x86.x64	MSVC v142 - VS 2019 C++ x64/x86 build tools (v14.20)	16.4.29511.114
Microsoft.VisualStudio.Component.VC.14.20.x86.x64.Spectre	MSVC v142 - VS 2019 C++ x64/x86 Spectre-mitigated libs (v14.20)	16.4.29511.114
Microsoft.VisualStudio.Component.VC.14.21.ARM	MSVC v142 - VS 2019 C++ ARM build tools (v14.21)	16.3.29207.166
Microsoft.VisualStudio.Component.VC.14.21.ARM.Spectre	MSVC v142 - VS 2019 C++ ARM Spectre-mitigated libs (v14.21)	16.4.29429.68
Microsoft.VisualStudio.Component.VC.14.21.ARM64	MSVC v142 - VS 2019 C++ ARM64 build tools (v14.21)	16.3.29207.166
Microsoft.VisualStudio.Component.VC.14.21.ARM64.Spectre	MSVC v142 - VS 2019 C++ ARM64 Spectre-mitigated libs (v14.21)	16.4.29429.68
Microsoft.VisualStudio.Component.VC.14.21.ATL	C++ v14.21 ATL for v142 build tools (x86 & x64)	16.2.29019.55
Microsoft.VisualStudio.Component.VC.14.21.ATL.ARM	C++ v14.21 ATL for v142 build tools (ARM)	16.2.29019.55
Microsoft.VisualStudio.Component.VC.14.21.ATL.ARM.Spectre	C++ v14.21 ATL for v142 build tools with Spectre Mitigations (ARM)	16.4.29429.68
Microsoft.VisualStudio.Component.VC.14.21.ATL.ARM64	C++ v14.21 ATL for v142 build tools (ARM64)	16.2.29019.55
Microsoft.VisualStudio.Component.VC.14.21.ATL.ARM64.Spectre	C++ v14.21 ATL for v142 build tools with Spectre Mitigations (ARM64)	16.4.29429.68
Microsoft.VisualStudio.Component.VC.14.21.ATL.Spectre	C++ v14.21 ATL for v142 build tools with Spectre Mitigations (x86 & x64)	16.4.29429.68
Microsoft.VisualStudio.Component.VC.14.21.CLI.Support	C++/CLI support for v142 build tools (14.21)	16.3.29207.166

Component ID	Name	Version
Microsoft.VisualStudio.Component.VC.14.21.MFC	C++ v14.21 MFC for v142 build tools (x86 & x64)	16.2.29019.55
Microsoft.VisualStudio.Component.VC.14.21.MFC.ARM	C++ v14.21 MFC for v142 build tools (ARM)	16.2.29019.55
Microsoft.VisualStudio.Component.VC.14.21.MFC.ARM.Spectre	C++ v14.21 MFC for v142 build tools with Spectre Mitigations (ARM)	16.4.29429.68
Microsoft.VisualStudio.Component.VC.14.21.MFC.ARM64	C++ v14.21 MFC for v142 build tools (ARM64)	16.2.29019.55
Microsoft.VisualStudio.Component.VC.14.21.MFC.ARM64.Spectre	C++ v14.21 MFC for v142 build tools with Spectre Mitigations (ARM64)	16.4.29429.68
Microsoft.VisualStudio.Component.VC.14.21.MFC.Spectre	C++ v14.21 MFC for v142 build tools with Spectre Mitigations (x86 & x64)	16.4.29429.68
Microsoft.VisualStudio.Component.VC.14.21.x86.x64	MSVC v142 - VS 2019 C++ x64/x86 build tools (v14.21)	16.3.29207.166
Microsoft.VisualStudio.Component.VC.14.21.x86.x64.Spectre	MSVC v142 - VS 2019 C++ x64/x86 Spectre-mitigated libs (v14.21)	16.4.29429.68
Microsoft.VisualStudio.Component.VC.14.22.ARM	MSVC v142 - VS 2019 C++ ARM build tools (v14.22)	16.4.29313.120
Microsoft.VisualStudio.Component.VC.14.22.ARM.Spectre	MSVC v142 - VS 2019 C++ ARM Spectre-mitigated libs (v14.22)	16.4.29429.68
Microsoft.VisualStudio.Component.VC.14.22.ARM64	MSVC v142 - VS 2019 C++ ARM64 build tools (v14.22)	16.4.29313.120
Microsoft.VisualStudio.Component.VC.14.22.ARM64.Spectre	MSVC v142 - VS 2019 C++ ARM64 Spectre-mitigated libs (v14.22)	16.4.29429.68
Microsoft.VisualStudio.Component.VC.14.22.ATL	C++ v14.22 ATL for v142 build tools (x86 & x64)	16.4.29313.120
Microsoft.VisualStudio.Component.VC.14.22.ATL.ARM	C++ v14.22 ATL for v142 build tools (ARM)	16.4.29313.120
Microsoft.VisualStudio.Component.VC.14.22.ATL.ARM.Spectre	C++ v14.22 ATL for v142 build tools with Spectre Mitigations (ARM)	16.4.29429.68
Microsoft.VisualStudio.Component.VC.14.22.ATL.ARM64	C++ v14.22 ATL for v142 build tools (ARM64)	16.4.29313.120
Microsoft.VisualStudio.Component.VC.14.22.ATL.ARM64.Spectre	C++ v14.22 ATL for v142 build tools with Spectre Mitigations (ARM64)	16.4.29429.68
Microsoft.VisualStudio.Component.VC.14.22.ATL.Spectre	C++ v14.22 ATL for v142 build tools with Spectre Mitigations (x86 & x64)	16.4.29429.68

Component ID	Name	Version
Microsoft.VisualStudio.Component.VC.14.22.CLI.Support	C++/CLI support for v142 build tools (14.22)	16.4.29313.120
Microsoft.VisualStudio.Component.VC.14.22.MFC	C++ v14.22 MFC for v142 build tools (x86 & x64)	16.4.29313.120
Microsoft.VisualStudio.Component.VC.14.22.MFC.ARM	C++ v14.22 MFC for v142 build tools (ARM)	16.4.29313.120
Microsoft.VisualStudio.Component.VC.14.22.MFC.ARM.Spectre	C++ v14.22 MFC for v142 build tools with Spectre Mitigations (ARM)	16.4.29429.68
Microsoft.VisualStudio.Component.VC.14.22.MFC.ARM64	C++ v14.22 MFC for v142 build tools (ARM64)	16.4.29313.120
Microsoft.VisualStudio.Component.VC.14.22.MFC.ARM64.Spectre	C++ v14.22 MFC for v142 build tools with Spectre Mitigations (ARM64)	16.4.29429.68
Microsoft.VisualStudio.Component.VC.14.22.MFC.Spectre	C++ v14.22 MFC for v142 build tools with Spectre Mitigations (x86 & x64)	16.4.29429.68
Microsoft.VisualStudio.Component.VC.14.22.x86.x64	MSVC v142 - VS 2019 C++ x64/x86 build tools (v14.22)	16.4.29313.120
Microsoft.VisualStudio.Component.VC.14.22.x86.x64.Spectre	MSVC v142 - VS 2019 C++ x64/x86 Spectre-mitigated libs (v14.22)	16.4.29429.68
Microsoft.VisualStudio.Component.VC.14.23.ARM	MSVC v142 - VS 2019 C++ ARM build tools (v14.23)	16.4.29429.68
Microsoft.VisualStudio.Component.VC.14.23.ARM.Spectre	MSVC v142 - VS 2019 C++ ARM Spectre-mitigated libs (v14.23)	16.4.29429.68
Microsoft.VisualStudio.Component.VC.14.23.ARM64	MSVC v142 - VS 2019 C++ ARM64 build tools (v14.23)	16.4.29429.68
Microsoft.VisualStudio.Component.VC.14.23.ARM64.Spectre	MSVC v142 - VS 2019 C++ ARM64 Spectre-mitigated libs (v14.23)	16.4.29429.68
Microsoft.VisualStudio.Component.VC.14.23.ATL	C++ v14.23 ATL for v142 build tools (x86 & x64)	16.4.29429.68
Microsoft.VisualStudio.Component.VC.14.23.ATL.ARM	C++ v14.23 ATL for v142 build tools (ARM)	16.4.29429.68
Microsoft.VisualStudio.Component.VC.14.23.ATL.ARM.Spectre	C++ v14.23 ATL for v142 build tools with Spectre Mitigations (ARM)	16.4.29429.68
Microsoft.VisualStudio.Component.VC.14.23.ATL.ARM64	C++ v14.23 ATL for v142 build tools (ARM64)	16.4.29429.68
Microsoft.VisualStudio.Component.VC.14.23.ATL.ARM64.Spectre	C++ v14.23 ATL for v142 build tools with Spectre Mitigations (ARM64)	16.4.29429.68

Component ID	Name	Version
Microsoft.VisualStudio.Component.VC.14.23.ATL.Spectre	C++ v14.23 ATL for v142 build tools with Spectre Mitigations (x86 & x64)	16.4.29429.68
Microsoft.VisualStudio.Component.VC.14.23.CLI.Support	C++/CLI support for v142 build tools (14.23)	16.4.29409.204
Microsoft.VisualStudio.Component.VC.14.23.MFC	C++ v14.23 MFC for v142 build tools (x86 & x64)	16.4.29429.68
Microsoft.VisualStudio.Component.VC.14.23.MFC.ARM	C++ v14.23 MFC for v142 build tools (ARM)	16.4.29429.68
Microsoft.VisualStudio.Component.VC.14.23.MFC.ARM.Spectre	C++ v14.23 MFC for v142 build tools with Spectre Mitigations (ARM)	16.4.29429.68
Microsoft.VisualStudio.Component.VC.14.23.MFC.ARM64	C++ v14.23 MFC for v142 build tools (ARM64)	16.4.29429.68
Microsoft.VisualStudio.Component.VC.14.23.MFC.ARM64.Spectre	C++ v14.23 MFC for v142 build tools with Spectre Mitigations (ARM64)	16.4.29429.68
Microsoft.VisualStudio.Component.VC.14.23.MFC.Spectre	C++ v14.23 MFC for v142 build tools with Spectre Mitigations (x86 & x64)	16.4.29429.68
Microsoft.VisualStudio.Component.VC.14.23.x86.x64	MSVC v142 - VS 2019 C++ x64/x86 build tools (v14.23)	16.4.29429.68
Microsoft.VisualStudio.Component.VC.14.23.x86.x64.Spectre	MSVC v142 - VS 2019 C++ x64/x86 Spectre-mitigated libs (v14.23)	16.4.29429.68
Microsoft.VisualStudio.Component.VC.ATL.ARM	C++ ATL for latest v142 build tools (ARM)	16.4.29313.120
Microsoft.VisualStudio.Component.VC.ATL.ARM.Spectre	C++ ATL for latest v142 build tools with Spectre Mitigations (ARM)	16.4.29429.68
Microsoft.VisualStudio.Component.VC.ATL.ARM64	C++ ATL for latest v142 build tools (ARM64)	16.4.29313.120
Microsoft.VisualStudio.Component.VC.ATL.ARM64.Spectre	C++ ATL for latest v142 build tools with Spectre Mitigations (ARM64)	16.4.29429.68
Microsoft.VisualStudio.Component.VC.ATL.Specre	C++ ATL for latest v142 build tools with Spectre Mitigations (x86 & x64)	16.4.29429.68
Microsoft.VisualStudio.Component.VC.ATLMFC.Specre	C++ MFC for latest v142 build tools with Spectre Mitigations (x86 & x64)	16.4.29429.68
Microsoft.VisualStudio.Component.VC.MFC.ARM	C++ MFC for latest v142 build tools (ARM)	16.4.29313.120
Microsoft.VisualStudio.Component.VC.MFC.ARM.Specre	C++ MFC for latest v142 build tools with Spectre Mitigations (ARM)	16.4.29429.68

Component ID	Name	Version
Microsoft.VisualStudio.Component.VC.MFC.ARM64	C++ MFC for latest v142 build tools (ARM64)	16.4.29313.120
Microsoft.VisualStudio.Component.VC.MFC.ARM64.Spectre	C++ MFC for latest v142 build tools with Spectre Mitigations (ARM64)	16.4.29429.68
Microsoft.VisualStudio.Component.VC.Redist.MSM	C++ 2019 Redistributable MSMs	16.4.29429.68
Microsoft.VisualStudio.Component.VC.RuntimeTools.ARM.Spectre	MSVC v142 - VS 2019 C++ ARM Spectre-mitigated libs (v14.24)	16.4.29429.68
Microsoft.VisualStudio.Component.VC.RuntimeTools.ARM64.Spectre	MSVC v142 - VS 2019 C++ ARM64 Spectre-mitigated libs (v14.24)	16.4.29429.68
Microsoft.VisualStudio.Component.VC.RuntimeTools.x86.x64.Spectre	MSVC v142 - VS 2019 C++ x64/x86 Spectre-mitigated libs (v14.24)	16.4.29429.68
Microsoft.VisualStudio.Component.VC.v141.ARM.Spectre	MSVC v141 - VS 2017 C++ ARM Spectre-mitigated libs (v14.16)	16.4.29429.68
Microsoft.VisualStudio.Component.VC.v141.ARM64.Spectre	MSVC v141 - VS 2017 C++ ARM64 Spectre-mitigated libs (v14.16)	16.4.29429.68
Microsoft.VisualStudio.Component.VC.v141.ATL	C++ ATL for v141 build tools (x86 & x64)	16.0.28625.61
Microsoft.VisualStudio.Component.VC.v141.ATL.ARM	C++ ATL for v141 build tools (ARM)	16.0.28625.61
Microsoft.VisualStudio.Component.VC.v141.ATL.ARM.Spectre	C++ ATL for v141 build tools with Spectre Mitigations (ARM)	16.0.28625.61
Microsoft.VisualStudio.Component.VC.v141.ATL.ARM64	C++ ATL for v141 build tools (ARM64)	16.0.28625.61
Microsoft.VisualStudio.Component.VC.v141.ATL.ARM64.Spectre	C++ ATL for v141 build tools with Spectre Mitigations (ARM64)	16.0.28625.61
Microsoft.VisualStudio.Component.VC.v141.ATL.Spectre	C++ ATL for v141 build tools with Spectre Mitigations (x86 & x64)	16.0.28625.61
Microsoft.VisualStudio.Component.VC.v141.CLI.Support	C++/CLI support for v141 build tools (14.16)	16.3.29207.166
Microsoft.VisualStudio.Component.VC.v141.MFC	C++ MFC for v141 build tools (x86 & x64)	16.0.28625.61
Microsoft.VisualStudio.Component.VC.v141.MFC.ARM	C++ MFC for v141 build tools (ARM)	16.2.28915.88
Microsoft.VisualStudio.Component.VC.v141.MFC.ARM.Spectre	C++ MFC for v141 build tools with Spectre Mitigations (ARM)	16.0.28625.61

Component ID	Name	Version
Microsoft.VisualStudio.Component.VC.v141.MFC.ARM64	C++ MFC for v141 build tools (ARM64)	16.0.28625.61
Microsoft.VisualStudio.Component.VC.v141.MFC.ARM64.Spectre	C++ MFC for v141 build tools with Spectre Mitigations (ARM64)	16.0.28625.61
Microsoft.VisualStudio.Component.VC.v141.MFC.Spectre	C++ MFC for v141 build tools with Spectre Mitigations (x86 & x64)	16.0.28625.61
Microsoft.VisualStudio.Component.VC.v141.x86.x64.Spectre	MSVC v141 - VS 2017 C++ x64/x86 Spectre-mitigated libs (v14.16)	16.4.29429.68
Microsoft.VisualStudio.Component.VisualStudioData	Data sources and service references	16.0.28707.177
Microsoft.VisualStudio.Component.WinXP	C++ Windows XP Support for VS 2017 (v141) tools [Deprecated]	16.1.28811.260
Microsoft.VisualStudio.Web.Mvc4.ComponentGroup	ASP.NET MVC 4	16.1.28810.153

Get support

Sometimes, things can go wrong. If your Visual Studio installation fails, see [Troubleshoot Visual Studio installation and upgrade issues](#) for step-by-step guidance.

We also offer a [live chat](#) (English only) support option for installation-related issues.

Here are a few more support options:

- Report product issues to us via the [Report a Problem](#) tool that appears both in the Visual Studio Installer and in the Visual Studio IDE.
- Suggest a feature, track product issues, and find answers in the [Visual Studio Developer Community](#).
- Use your [GitHub](#) account to talk to us and other Visual Studio developers in the [Visual Studio conversation in the Gitter community](#).

See also

- [Visual Studio workload and component IDs](#)
- [Visual Studio administrator guide](#)
- [Use command-line parameters to install Visual Studio](#)
 - [Command-line parameter examples](#)
- [Create an offline installation of Visual Studio](#)

Visual Studio Professional component directory

12/4/2019 • 67 minutes to read • [Edit Online](#)

The tables on this page list the IDs that you can use to install Visual Studio from the command line, or that you can specify as a dependency in a VSIX manifest. We'll add additional components when we release updates to Visual Studio.

Also note the following about the page:

- Each workload has its own section, followed by the workload ID and a table of the components that are available for the workload.
- By default, the **Required** components will be installed when you install the workload.
- If you choose to, you can also install the **Recommended** and **Optional** components.
- We've also added a section that lists the additional components that are not affiliated with any workload.

When you set dependencies in your VSIX manifest, you must specify Component IDs only. Use the tables on this page to determine the minimum component dependencies. In some scenarios, this might mean that you specify only one component from a workload. In other scenarios, it might mean that you specify multiple components from a single workload or multiple components from multiple workloads. For more information, see the [How to: Migrate extensibility projects to Visual Studio 2017](#) page.

For more information about how to use these IDs, see the [Use command-line parameters to install Visual Studio 2017](#) page. And, for a list of workload and component IDs for other products, see the [Visual Studio workload and component IDs](#) page.

Visual Studio core editor (included with Visual Studio Professional 2017)

ID: Microsoft.VisualStudio.Workload.CoreEditor

Description: The Visual Studio core shell experience, including syntax-aware code editing, source code control and work item management.

Components included by this workload

COMPONENT ID	NAME	VERSION	DEPENDENCY TYPE
Microsoft.VisualStudio.Component.CoreEditor	Visual Studio core editor	15.8.27729.1	Required
Microsoft.VisualStudio.Component.StartPageExperiment.Cpp	Visual Studio Start Page for C++ Users	15.0.27128.1	Optional

Azure development

ID: Microsoft.VisualStudio.Workload.Azure

Description: Azure SDKs, tools, and projects for developing cloud apps, creating resources, and building Containers including Docker support.

Components included by this workload

Component ID	Name	Version	Dependency Type
Component.Microsoft.VisualStudio.RazorExtension	Razor Language Services	15.0.26720.2	Required
Component.Microsoft.VisualStudio.Web.AzureFunctions	Microsoft Azure WebJobs Tools	15.7.27617.1	Required
Component.Microsoft.Web.LibraryManager	Library Manager	15.8.27705.0	Required
Component.WebSocket	WebSocket4Net	15.0.26606.0	Required
Microsoft.Component.ClickOnce	ClickOnce Publishing	15.8.27825.0	Required
Microsoft.Component.MSBuild	MSBuild	15.7.27520.0	Required
Microsoft.Component.NetFX.Core.Runtime	.NET Core runtime	15.0.26208.0	Required
Microsoft.Net.Component.4.5.2.TargetingPack	.NET Framework 4.5.2 targeting pack	15.6.27406.0	Required
Microsoft.Net.Component.4.5.TargetingPack	.NET Framework 4.5 targeting pack	15.6.27406.0	Required
Microsoft.Net.Component.4.6.1.SDK	.NET Framework 4.6.1 SDK	15.6.27406.0	Required
Microsoft.Net.Component.4.6.1.TargetingPack	.NET Framework 4.6.1 targeting pack	15.6.27406.0	Required
Microsoft.Net.ComponentGroup.DevelopmentPrerequisites	.NET Framework 4.6.1 development tools	15.8.27825.0	Required
Microsoft.Net.Core.Component.SDK.2.1	.NET Core 2.1 development tools	15.8.27924.0	Required
Microsoft.NetCore.ComponentGroup.DevelopmentTools.2.1	.NET Core 2.1 development tools	15.8.27924.0	Required
Microsoft.NetCore.ComponentGroup.Web.2.1	.NET Core 2.1 development tools	15.8.27924.0	Required
Microsoft.VisualStudio.Component.Azure.AuthoringTools	Azure Authoring Tools	15.9.28307.421	Required
Microsoft.VisualStudio.Component.Azure.ClientLibs	Azure libraries for .NET	15.0.26208.0	Required

Component ID	Name	Version	Dependency Type
Microsoft.VisualStudio.Component.Azure.Compute Emulator	Azure Compute Emulator	15.9.28307.421	Required
Microsoft.VisualStudio.Component.Azure.Storage.Emulator	Azure Storage Emulator	15.9.28125.51	Required
Microsoft.VisualStudio.Component.CloudExplorer	Cloud Explorer	15.9.28230.55	Required
Microsoft.VisualStudio.Component.Common.Azure.Tools	Connectivity and publishing tools	15.9.28107.0	Required
Microsoft.VisualStudio.Component.DockerTools	Container development tools	15.8.27906.1	Required
Microsoft.VisualStudio.Component.DockerTools.BuildTools	Container development tools - Build Tools	15.7.27617.1	Required
Microsoft.VisualStudio.Component.FSharp	F# language support	15.8.27825.0	Required
Microsoft.VisualStudio.Component.FSharp.WebTemplates	F# language support for web projects	15.9.28307.421	Required
Microsoft.VisualStudio.Component.IISExpress	IIS Express	15.0.26208.0	Required
Microsoft.VisualStudio.Component.JavaScript.Diagnostics	JavaScript diagnostics	15.8.27729.1	Required
Microsoft.VisualStudio.Component.JavaScript.TypeScript	JavaScript and TypeScript language support	15.9.28125.51	Required
Microsoft.VisualStudio.Component.ManagedDesktop.Core	Managed Desktop Workload Core	15.8.27729.1	Required
Microsoft.VisualStudio.Component.NuGet	NuGet package manager	15.9.28016.0	Required
Microsoft.VisualStudio.Component.PortableLibrary	.NET Portable Library targeting pack	15.6.27309.0	Required
Microsoft.VisualStudio.Component.Roslyn.Compiler	C# and Visual Basic Roslyn compilers	15.6.27309.0	Required
Microsoft.VisualStudio.Component.Roslyn.LanguageServices	C# and Visual Basic	15.8.27729.1	Required

Component ID	Name	Version	Dependency Type
Microsoft.VisualStudio.Component.SQL.ADAL	SQL ADAL runtime	15.6.27406.0	Required
Microsoft.VisualStudio.Component.SQL.CLR	CLR data types for SQL Server	15.0.26208.0	Required
Microsoft.VisualStudio.Component.SQL.CMDUtils	SQL Server Command Line Utilities	15.0.26208.0	Required
Microsoft.VisualStudio.Component.SQL.DataSources	Data sources for SQL Server support	15.0.26621.2	Required
Microsoft.VisualStudio.Component.SQL.LocalDB.Runtime	SQL Server Express 2016 LocalDB	15.7.27617.1	Required
Microsoft.VisualStudio.Component.SQL.NCLI	SQL Server Native Client	15.0.26208.0	Required
Microsoft.VisualStudio.Component.SQL.SSDT	SQL Server Data Tools	15.9.28107.0	Required
Microsoft.VisualStudio.Component.Static.Analysis.Tools	Static analysis tools	15.0.26208.0	Required
Microsoft.VisualStudio.Component.TextTemplating	Text Template Transformation	15.0.26208.0	Required
Microsoft.VisualStudio.Component.TypeScript.3.1	TypeScript 3.1 SDK	15.0.28218.60	Required
Microsoft.VisualStudio.Component.VisualStudioData	Data sources and service references	15.6.27406.0	Required
Microsoft.VisualStudio.Component.Web	ASP.NET and web development tools	15.8.27825.0	Required
Microsoft.VisualStudio.ComponentGroup.Azure.Prerequisites	Azure development prerequisites	15.9.28107.0	Required
Microsoft.VisualStudio.ComponentGroup.AzureFunctions	Microsoft Azure WebJobs Tools	15.7.27617.1	Required
Microsoft.VisualStudio.ComponentGroup.Web	ASP.NET and web development tools prerequisites	15.9.28219.51	Required
Microsoft.VisualStudio.ComponentGroup.WebToolsExtensions	ASP.NET and web development	15.8.27825.0	Required
Microsoft.Component.Azure.DataLake.Tools	Azure Data Lake and Stream Analytics Tools	15.9.28107.0	Recommended

Component ID	Name	Version	Dependency Type
Microsoft.Net.Component.4.5.1.TargetingPack	.NET Framework 4.5.1 targeting pack	15.6.27406.0	Recommended
Microsoft.Net.Component.4.6.TargetingPack	.NET Framework 4.6 targeting pack	15.6.27406.0	Recommended
Microsoft.Net.Component.4.TargetingPack	.NET Framework 4 targeting pack	15.6.27406.0	Recommended
Microsoft.ComponentGroup.TargetingPacks.Common	.NET Framework 4 – 4.6 development tools	15.6.27406.0	Recommended
Microsoft.VisualStudio.Component.AspNet45	Advanced ASP.NET features	15.7.27625.0	Recommended
Microsoft.VisualStudio.Component.Azure.MobileAppsSdk	Azure Mobile Apps SDK	15.7.27625.0	Recommended
Microsoft.VisualStudio.Component.Azure.ResourceManager.Tools	Azure Resource Manager core tools	15.9.28107.0	Recommended
Microsoft.VisualStudio.Component.Azure.ServiceFabric.Tools	Service Fabric Tools	15.8.27825.0	Recommended
Microsoft.VisualStudio.Component.Azure.Waverton	Azure Cloud Services core tools	15.9.28107.0	Recommended
Microsoft.VisualStudio.Component.Azure.Waverton.BuildTools	Azure Cloud Services build tools	15.7.27617.1	Recommended
Microsoft.VisualStudio.Component.DiagnosticTools	.NET profiling tools	15.8.27729.1	Recommended
Microsoft.VisualStudio.Component.WebDeploy	Web Deploy	15.8.27729.1	Recommended
Microsoft.VisualStudio.ComponentGroup.Azure.CloudServices	Azure Cloud Services tools	15.0.26504.0	Recommended
Microsoft.VisualStudio.ComponentGroup.Azure.ResourceManager.Tools	Azure Resource Manager tools	15.0.27005.2	Recommended
Microsoft.Net.Component.4.6.2.SDK	.NET Framework 4.6.2 SDK	15.6.27406.0	Optional
Microsoft.Net.Component.4.6.2.TargetingPack	.NET Framework 4.6.2 targeting pack	15.6.27406.0	Optional

Component ID	Name	Version	Dependency Type
Microsoft.Net.Component.4.7.1.SDK	.NET Framework 4.7.1 SDK	15.6.27406.0	Optional
Microsoft.Net.Component.4.7.1.TargetingPack	.NET Framework 4.7.1 targeting pack	15.6.27406.0	Optional
Microsoft.Net.Component.4.7.2.SDK	.NET Framework 4.7.2 SDK	15.8.27825.0	Optional
Microsoft.Net.Component.4.7.2.TargetingPack	.NET Framework 4.7.2 targeting pack	15.8.27825.0	Optional
Microsoft.Net.Component.4.7.SDK	.NET Framework 4.7 SDK	15.6.27406.0	Optional
Microsoft.Net.Component.4.7.TargetingPack	.NET Framework 4.7 targeting pack	15.6.27406.0	Optional
Microsoft.Net.ComponentGroup.4.6.2.DeveloperTools	.NET Framework 4.6.2 development tools	15.6.27406.0	Optional
Microsoft.Net.ComponentGroup.4.7.1.DeveloperTools	.NET Framework 4.7.1 development tools	15.6.27406.0	Optional
Microsoft.Net.ComponentGroup.4.7.2.DeveloperTools	.NET Framework 4.7.2 development tools	15.8.27825.0	Optional
Microsoft.Net.ComponentGroup.4.7.DeveloperTools	.NET Framework 4.7 development tools	15.6.27406.0	Optional
Microsoft.Net.Core.Component.SDK	.NET Core 2.0 development tools	15.6.27406.0	Optional
Microsoft.Net.Core.Component.SDK.1x	.NET Core 1.0 - 1.1 development tools	15.6.27406.0	Optional
Microsoft.NetCore.1x.ComponentGroup.Web	.NET Core 1.0 - 1.1 development tools for Web	15.6.27406.0	Optional
Microsoft.NetCore.ComponentGroup.DevelopmentTools	.NET Core 2.0 development tools	15.8.27729.1	Optional
Microsoft.NetCore.ComponentGroup.Web	.NET Core 2.0 development tools	15.7.27625.0	Optional
Microsoft.VisualStudio.Component.Azure.Storage.AzCopy	Azure Storage AzCopy	15.0.26906.1	Optional
Microsoft.VisualStudio.Component.Wcf.Tooling	Windows Communication Foundation	15.8.27924.0	Optional

Data storage and processing

ID: Microsoft.VisualStudio.Workload.Data

Description: Connect, develop, and test data solutions with SQL Server, Azure Data Lake, or Hadoop.

Components included by this workload

Component ID	Name	Version	Dependency Type
Component.Microsoft.VisualStudio.RazorExtension	Razor Language Services	15.0.26720.2	Recommended
Component.Microsoft.Web.LibraryManager	Library Manager	15.8.27705.0	Recommended
Component.Redgate.SQLSearch.VSExtension	Redgate SQL Search	3.1.7.2062	Recommended
Component.WebSocket	WebSocket4Net	15.0.26606.0	Recommended
Microsoft.Component.Azure.DataLake.Tools	Azure Data Lake and Stream Analytics Tools	15.9.28107.0	Recommended
Microsoft.Component.ClickOnce	ClickOnce Publishing	15.8.27825.0	Recommended
Microsoft.Component.MSBuild	MSBuild	15.7.27520.0	Recommended
Microsoft.Net.Component.4.5.1.TargetingPack	.NET Framework 4.5.1 targeting pack	15.6.27406.0	Recommended
Microsoft.Net.Component.4.5.2.TargetingPack	.NET Framework 4.5.2 targeting pack	15.6.27406.0	Recommended
Microsoft.Net.Component.4.5.TargetingPack	.NET Framework 4.5 targeting pack	15.6.27406.0	Recommended
Microsoft.Net.Component.4.6.1.SDK	.NET Framework 4.6.1 SDK	15.6.27406.0	Recommended
Microsoft.Net.Component.4.6.1.TargetingPack	.NET Framework 4.6.1 targeting pack	15.6.27406.0	Recommended
Microsoft.Net.Component.4.6.TargetingPack	.NET Framework 4.6 targeting pack	15.6.27406.0	Recommended
Microsoft.Net.Component.4.TargetingPack	.NET Framework 4 targeting pack	15.6.27406.0	Recommended
Microsoft.Net.ComponentGroup.DevelopmentPrerequisites	.NET Framework 4.6.1 development tools	15.8.27825.0	Recommended
Microsoft.Net.ComponentGroup.TargetingPacks.Common	.NET Framework 4 – 4.6 development tools	15.6.27406.0	Recommended

Component ID	Name	Version	Dependency Type
Microsoft.Net.Core.Component.SDK.2.1	.NET Core 2.1 development tools	15.8.27924.0	Recommended
Microsoft.VisualStudio.Component.Azure.AuthoringTools	Azure Authoring Tools	15.9.28307.421	Recommended
Microsoft.VisualStudio.Component.Azure.ClientLibs	Azure libraries for .NET	15.0.26208.0	Recommended
Microsoft.VisualStudio.Component.Azure.Compute.Emulator	Azure Compute Emulator	15.9.28307.421	Recommended
Microsoft.VisualStudio.Component.Azure.Storage.Emulator	Azure Storage Emulator	15.9.28125.51	Recommended
Microsoft.VisualStudio.Component.Azure.Waverton	Azure Cloud Services core tools	15.9.28107.0	Recommended
Microsoft.VisualStudio.Component.Azure.Waverton.BuildTools	Azure Cloud Services build tools	15.7.27617.1	Recommended
Microsoft.VisualStudio.Component.CloudExplorer	Cloud Explorer	15.9.28230.55	Recommended
Microsoft.VisualStudio.Component.Common.Azure.Tools	Connectivity and publishing tools	15.9.28107.0	Recommended
Microsoft.VisualStudio.Component.DockerTools	Container development tools	15.8.27906.1	Recommended
Microsoft.VisualStudio.Component.DockerTools.BuildTools	Container development tools - Build Tools	15.7.27617.1	Recommended
Microsoft.VisualStudio.Component.IISExpress	IIS Express	15.0.26208.0	Recommended
Microsoft.VisualStudio.Component.JavaScript.Diagnostics	JavaScript diagnostics	15.8.27729.1	Recommended
Microsoft.VisualStudio.Component.JavaScript.TypeScript	JavaScript and TypeScript language support	15.9.28125.51	Recommended
Microsoft.VisualStudio.Component.ManagedDesktop.Core	Managed Desktop Workload Core	15.8.27729.1	Recommended
Microsoft.VisualStudio.Component.NuGet	NuGet package manager	15.9.28016.0	Recommended

Component ID	Name	Version	Dependency Type
Microsoft.VisualStudio.Component.PortableLibrary	.NET Portable Library targeting pack	15.6.27309.0	Recommended
Microsoft.VisualStudio.Component.Roslyn.Compiler	C# and Visual Basic Roslyn compilers	15.6.27309.0	Recommended
Microsoft.VisualStudio.Component.Roslyn.LanguageServices	C# and Visual Basic	15.8.27729.1	Recommended
Microsoft.VisualStudio.Component.SQL.ADAL	SQL ADAL runtime	15.6.27406.0	Recommended
Microsoft.VisualStudio.Component.SQL.CLR	CLR data types for SQL Server	15.0.26208.0	Recommended
Microsoft.VisualStudio.Component.SQL.CMDUtils	SQL Server Command Line Utilities	15.0.26208.0	Recommended
Microsoft.VisualStudio.Component.SQL.DataSources	Data sources for SQL Server support	15.0.26621.2	Recommended
Microsoft.VisualStudio.Component.SQL.LocalDB.Runtime	SQL Server Express 2016 LocalDB	15.7.27617.1	Recommended
Microsoft.VisualStudio.Component.SQL.NCLI	SQL Server Native Client	15.0.26208.0	Recommended
Microsoft.VisualStudio.Component.SQL.SSDT	SQL Server Data Tools	15.9.28107.0	Recommended
Microsoft.VisualStudio.Component.Static.Analysis.Tools	Static analysis tools	15.0.26208.0	Recommended
Microsoft.VisualStudio.Component.TextTemplating	Text Template Transformation	15.0.26208.0	Recommended
Microsoft.VisualStudio.Component.TypeScript.3.1	TypeScript 3.1 SDK	15.0.28218.60	Recommended
Microsoft.VisualStudio.Component.VisualStudioData	Data sources and service references	15.6.27406.0	Recommended
Microsoft.VisualStudio.Component.Web	ASP.NET and web development tools	15.8.27825.0	Recommended
Microsoft.VisualStudio.ComponentGroup.Web	ASP.NET and web development tools prerequisites	15.9.28219.51	Recommended
Microsoft.VisualStudio.ComponentGroup.WebToolsExtensions	ASP.NET and web development	15.8.27825.0	Recommended

Component ID	Name	Version	Dependency Type
Microsoft.VisualStudio.Component.FSharp.Desktop	F# desktop language support	15.8.27825.0	Optional

Data science and analytical applications

ID: Microsoft.VisualStudio.Workload.DataScience

Description: Languages and tooling for creating data science applications, including Python, R and F#.

Components included by this workload

Component ID	Name	Version	Dependency Type
Component.Anconda3.x64	Anaconda3 64-bit (5.2.0)	5.2.0	Recommended
Microsoft.Component.CookiecutterTools	Cookiecutter template support	15.0.26621.2	Recommended
Microsoft.Component.PythonTools	Python language support	15.0.26823.1	Recommended
Microsoft.Component.PythonTools.Web	Python web support	15.9.28107.0	Recommended
Microsoft.Net.Component.4.6.1.TargetingPack	.NET Framework 4.6.1 targeting pack	15.6.27406.0	Recommended
Microsoft.VisualStudio.Component.Common.Azure.Tools	Connectivity and publishing tools	15.9.28107.0	Recommended
Microsoft.VisualStudio.Component.FSharp.Desktop	F# desktop language support	15.8.27825.0	Recommended
Microsoft.VisualStudio.Component.JavaScript.TypeScript	JavaScript and TypeScript language support	15.9.28125.51	Recommended
Microsoft.VisualStudio.Component.NuGet	NuGet package manager	15.9.28016.0	Recommended
Microsoft.VisualStudio.Component.R.Open	Microsoft R Client (3.3.2)	15.6.27406.0	Recommended
Microsoft.VisualStudio.Component.RHost	Runtime support for R development tools	15.6.27406.0	Recommended
Microsoft.VisualStudio.Component.Roslyn.Compiler	C# and Visual Basic Roslyn compilers	15.6.27309.0	Recommended
Microsoft.VisualStudio.Component.Roslyn.LanguageServices	C# and Visual Basic	15.8.27729.1	Recommended
Microsoft.VisualStudio.Component.RTools	R language support	15.0.26919.1	Recommended

Component ID	Name	Version	Dependency Type
Microsoft.VisualStudio.Component.SQL CLR	CLR data types for SQL Server	15.0.26208.0	Recommended
Microsoft.VisualStudio.Component.Static.Analysis.Tools	Static analysis tools	15.0.26208.0	Recommended
Microsoft.VisualStudio.Component.TypeScript.3.1	TypeScript 3.1 SDK	15.0.28218.60	Recommended
Microsoft.VisualStudio.Component.VisualStudioData	Data sources and service references	15.6.27406.0	Recommended
Microsoft.VisualStudio.Component.WebDeploy	Web Deploy	15.8.27729.1	Recommended
Microsoft.VisualStudio.ComponentGroup.WebToolsExtensions	ASP.NET and web development	15.8.27825.0	Recommended
Component.Anaconda2.x64	Anaconda2 64-bit (5.2.0)	5.2.0	Optional
Component.Anaconda2.x86	Anaconda2 32-bit (5.2.0)	5.2.0	Optional
Component.Anaconda3.x86	Anaconda3 32-bit (5.2.0)	5.2.0	Optional
Microsoft.Component.VC.Runtime.UCRTSDK	Windows Universal CRT SDK	15.6.27309.0	Optional
Microsoft.ComponentGroup.PythonTools.NativeDevelopment	Python native development tools	15.9.28307.102	Optional
Microsoft.VisualStudio.Component.Graphics.Tools	Graphics debugger and GPU profiler for DirectX	15.6.27406.0	Optional
Microsoft.VisualStudio.Component.Graphics.Win81	Graphics Tools Windows 8.1 SDK	15.6.27406.0	Optional
Microsoft.VisualStudio.Component.VC.140	VC++ 2015.3 v14.00 (v140) toolset for desktop	15.7.27617.1	Optional
Microsoft.VisualStudio.Component.VC.Corelde	Visual Studio C++ core features	15.6.27406.0	Optional
Microsoft.VisualStudio.Component.VC.DiagnosticTools	C++ profiling tools	15.0.26823.1	Optional
Microsoft.VisualStudio.Component.VC.Tools.x86.x64	VC++ 2017 version 15.9 v14.16 latest v141 tools	15.9.28230.55	Optional
Microsoft.VisualStudio.Component.Windows10SDK	Windows Universal C Runtime	15.6.27406.0	Optional

Component ID	Name	Version	Dependency Type
Microsoft.VisualStudio.Component.Windows10SDK.17763	Windows 10 SDK (10.0.17763.0)	15.9.28307.102	Optional
Microsoft.VisualStudio.Component.Windows81SDK	Windows 8.1 SDK	15.6.27406.0	Optional

.NET desktop development

ID: Microsoft.VisualStudio.Workload.ManagedDesktop

Description: Build WPF, Windows Forms, and console applications using C#, Visual Basic, and F#.

Components included by this workload

Component ID	Name	Version	Dependency Type
Microsoft.Component.ClickOnce	ClickOnce Publishing	15.8.27825.0	Required
Microsoft.Component.MSBuild	MSBuild	15.7.27520.0	Required
Microsoft.Net.Component.4.6.1.SDK	.NET Framework 4.6.1 SDK	15.6.27406.0	Required
Microsoft.Net.Component.4.6.1.TargetingPack	.NET Framework 4.6.1 targeting pack	15.6.27406.0	Required
Microsoft.Net.ComponentGroup.DevelopmentPrerequisites	.NET Framework 4.6.1 development tools	15.8.27825.0	Required
Microsoft.VisualStudio.Component.ManagedDesktop.Core	Managed Desktop Workload Core	15.8.27729.1	Required
Microsoft.VisualStudio.Component.ManagedDesktop.Prerequisites	.NET desktop development tools	15.7.27625.0	Required
Microsoft.VisualStudio.Component.PortableLibrary	.NET Portable Library targeting pack	15.6.27309.0	Required
Microsoft.VisualStudio.Component.Roslyn.Compiler	C# and Visual Basic Roslyn compilers	15.6.27309.0	Required
Microsoft.VisualStudio.Component.Roslyn.LanguageServices	C# and Visual Basic	15.8.27729.1	Required
Microsoft.VisualStudio.Component.SQL.CLR	CLR data types for SQL Server	15.0.26208.0	Required

Component ID	Name	Version	Dependency Type
Microsoft.VisualStudio.Component.Static.Analysis.Tools	Static analysis tools	15.0.26208.0	Required
Microsoft.VisualStudio.Component.TextTemplating	Text Template Transformation	15.0.26208.0	Required
Microsoft.VisualStudio.Component.VisualStudioData	Data sources and service references	15.6.27406.0	Required
Microsoft.ComponentGroup.Blend	Blend for Visual Studio	15.6.27406.0	Recommended
Microsoft.Net.Component.4.5.1.TargetingPack	.NET Framework 4.5.1 targeting pack	15.6.27406.0	Recommended
Microsoft.Net.Component.4.5.2.TargetingPack	.NET Framework 4.5.2 targeting pack	15.6.27406.0	Recommended
Microsoft.Net.Component.4.5.TargetingPack	.NET Framework 4.5 targeting pack	15.6.27406.0	Recommended
Microsoft.Net.Component.4.6.TargetingPack	.NET Framework 4.6 targeting pack	15.6.27406.0	Recommended
Microsoft.Net.Component.4.TargetingPack	.NET Framework 4 targeting pack	15.6.27406.0	Recommended
Microsoft.Net.ComponentGroup.TargetingPacks.Common	.NET Framework 4 – 4.6 development tools	15.6.27406.0	Recommended
Microsoft.VisualStudio.Component.Debugger.JustInTime	Just-In-Time debugger	15.0.27005.2	Recommended
Microsoft.VisualStudio.Component.EntityFramework	Entity Framework 6 tools	15.6.27406.0	Recommended
Component.Dotfuscator	PreEmptive Protection - Dotfuscator	15.0.26208.0	Optional
Component.Microsoft.VisualStudio.RazorExtension	Razor Language Services	15.0.26720.2	Optional
Component.Microsoft.Web.LibraryManager	Library Manager	15.8.27705.0	Optional
Component.WebSocket	WebSocket4Net	15.0.26606.0	Optional
Microsoft.Net.Component.4.6.2.SDK	.NET Framework 4.6.2 SDK	15.6.27406.0	Optional
Microsoft.Net.Component.4.6.2.TargetingPack	.NET Framework 4.6.2 targeting pack	15.6.27406.0	Optional

Component ID	Name	Version	Dependency Type
Microsoft.Net.Component.4.7.1.SDK	.NET Framework 4.7.1 SDK	15.6.27406.0	Optional
Microsoft.Net.Component.4.7.1.TargetingPack	.NET Framework 4.7.1 targeting pack	15.6.27406.0	Optional
Microsoft.Net.Component.4.7.2.SDK	.NET Framework 4.7.2 SDK	15.8.27825.0	Optional
Microsoft.Net.Component.4.7.2.TargetingPack	.NET Framework 4.7.2 targeting pack	15.8.27825.0	Optional
Microsoft.Net.Component.4.7.SDK	.NET Framework 4.7 SDK	15.6.27406.0	Optional
Microsoft.Net.Component.4.7.TargetingPack	.NET Framework 4.7 targeting pack	15.6.27406.0	Optional
Microsoft.Net.ComponentGroup.4.6.2.DeveloperTools	.NET Framework 4.6.2 development tools	15.6.27406.0	Optional
Microsoft.Net.ComponentGroup.4.7.1.DeveloperTools	.NET Framework 4.7.1 development tools	15.6.27406.0	Optional
Microsoft.Net.ComponentGroup.4.7.2.DeveloperTools	.NET Framework 4.7.2 development tools	15.8.27825.0	Optional
Microsoft.Net.ComponentGroup.4.7.DeveloperTools	.NET Framework 4.7 development tools	15.6.27406.0	Optional
Microsoft.Net.Core.Component.SDK	.NET Core 2.0 development tools	15.6.27406.0	Optional
Microsoft.Net.Core.Component.SDK.1x	.NET Core 1.0 - 1.1 development tools	15.6.27406.0	Optional
Microsoft.Net.Core.Component.SDK.2.1	.NET Core 2.1 development tools	15.8.27924.0	Optional
Microsoft.NetCore.ComponentGroup.DevelopmentTools	.NET Core 2.0 development tools	15.8.27729.1	Optional
Microsoft.NetCore.ComponentGroup.DevelopmentTools.2.1	.NET Core 2.1 development tools	15.8.27924.0	Optional
Microsoft.VisualStudio.Component.Common.Azure.Tools	Connectivity and publishing tools	15.9.28107.0	Optional
Microsoft.VisualStudio.Component.DockerTools	Container development tools	15.8.27906.1	Optional

Component ID	Name	Version	Dependency Type
Microsoft.VisualStudio.Component.DockerTools.BuildTools	Container development tools - Build Tools	15.7.27617.1	Optional
Microsoft.VisualStudio.Component.FSharp	F# language support	15.8.27825.0	Optional
Microsoft.VisualStudio.Component.FSharp.Desktop	F# desktop language support	15.8.27825.0	Optional
Microsoft.VisualStudio.Component.IIExpress	IIS Express	15.0.26208.0	Optional
Microsoft.VisualStudio.Component.JavaScript.Diagnostics	JavaScript diagnostics	15.8.27729.1	Optional
Microsoft.VisualStudio.Component.JavaScript.TypeScript	JavaScript and TypeScript language support	15.9.28125.51	Optional
Microsoft.VisualStudio.Component.NuGet	NuGet package manager	15.9.28016.0	Optional
Microsoft.VisualStudio.Component.SQL.ADAL	SQL ADAL runtime	15.6.27406.0	Optional
Microsoft.VisualStudio.Component.SQL.CMDUtils	SQL Server Command Line Utilities	15.0.26208.0	Optional
Microsoft.VisualStudio.Component.SQL.DataSources	Data sources for SQL Server support	15.0.26621.2	Optional
Microsoft.VisualStudio.Component.SQL.LocalDB.Runtime	SQL Server Express 2016 LocalDB	15.7.27617.1	Optional
Microsoft.VisualStudio.Component.SQL.NCLI	SQL Server Native Client	15.0.26208.0	Optional
Microsoft.VisualStudio.Component.SQL.SSDT	SQL Server Data Tools	15.9.28107.0	Optional
Microsoft.VisualStudio.Component.TypeScript.3.1	TypeScript 3.1 SDK	15.0.28218.60	Optional
Microsoft.VisualStudio.Component.Wcf.Tooling	Windows Communication Foundation	15.8.27924.0	Optional
Microsoft.VisualStudio.Component.Web	ASP.NET and web development tools	15.8.27825.0	Optional
Microsoft.VisualStudio.ComponentGroup.Web	ASP.NET and web development tools prerequisites	15.9.28219.51	Optional

Component ID	Name	Version	Dependency Type
Microsoft.VisualStudio.ComponentGroup.WebToolsExtensions	ASP.NET and web development	15.8.27825.0	Optional

Game development with Unity

ID: Microsoft.VisualStudio.Workload.ManagedGame

Description: Create 2D and 3D games with Unity, a powerful cross-platform development environment.

Components included by this workload

Component ID	Name	Version	Dependency Type
Microsoft.Net.Component.3.5.DeveloperTools	.NET Framework 3.5 development tools	15.6.27406.0	Required
Microsoft.Net.Component.4.7.1.TargetingPack	.NET Framework 4.7.1 targeting pack	15.6.27406.0	Required
Microsoft.VisualStudio.Component.NuGet	NuGet package manager	15.9.28016.0	Required
Microsoft.VisualStudio.Component.Roslyn.Compiler	C# and Visual Basic Roslyn compilers	15.6.27309.0	Required
Microsoft.VisualStudio.Component.Roslyn.LanguageServices	C# and Visual Basic	15.8.27729.1	Required
Microsoft.VisualStudio.Component.Static.Analysis.Tools	Static analysis tools	15.0.26208.0	Required
Microsoft.VisualStudio.Component.Unity	Visual Studio Tools for Unity	15.7.27617.1	Required
Component.UnityEngine.x64	Unity 2018.3 64-bit Editor	15.9.28307.271	Recommended
Component.UnityEngine.x86	Unity 5.6 32-bit Editor	15.6.27406.0	Recommended

Linux development with C++

ID: Microsoft.VisualStudio.Workload.NativeCrossPlat

Description: Create and debug applications running in a Linux environment.

Components included by this workload

Component ID	Name	Version	Dependency Type
Component.MDD.Linux	Visual C++ for Linux Development	15.6.27406.0	Required

Component ID	Name	Version	Dependency Type
Microsoft.VisualStudio.Component.VC.CoreLDE	Visual Studio C++ core features	15.6.27406.0	Required
Microsoft.VisualStudio.Component.Windows10SDK	Windows Universal C Runtime	15.6.27406.0	Required
Component.Linux.CMake	Visual C++ tools for CMake and Linux	15.9.28307.102	Recommended
Microsoft.VisualStudio.Component.Static.Analysis.Tools	Static analysis tools	15.0.26208.0	Recommended
Microsoft.VisualStudio.Component.VC.Tools.x86.x64	VC++ 2017 version 15.9 v14.16 latest v141 tools	15.9.28230.55	Recommended
Microsoft.VisualStudio.Component.Windows10SDK.17763	Windows 10 SDK (10.0.17763.0)	15.9.28307.102	Recommended
Microsoft.VisualStudio.ComponentGroup.WebToolsExtensions	ASP.NET and web development	15.8.27825.0	Recommended
Component.MDD.Linux.GCC.arm	Embedded and IoT Development	15.6.27309.0	Optional

Desktop development with C++

ID: Microsoft.VisualStudio.Workload.NativeDesktop

Description: Build Windows desktop applications using the Microsoft C++ toolset, ATL, or MFC.

Components included by this workload

COMPONENT ID	NAME	VERSION	DEPENDENCY TYPE
Microsoft.Component_MSBuild	MSBuild	15.7.27520.0	Required
Microsoft.VisualStudio.Component.Roslyn.Compiler	C# and Visual Basic Roslyn compilers	15.6.27309.0	Required
Microsoft.VisualStudio.Component.TextTemplating	Text Template Transformation	15.0.26208.0	Required
Microsoft.VisualStudio.Component.VC.CoreLDE	Visual Studio C++ core features	15.6.27406.0	Required
Microsoft.VisualStudio.Component.VC.Redist.14.Latest	Visual C++ 2017 Redistributable Update	15.6.27406.0	Required
Microsoft.VisualStudio.ComponentGroup.NativeDesktop.Core	Visual C++ core desktop features	15.8.27729.1	Required

Component ID	Name	Version	Dependency Type
Microsoft.VisualStudio.Component.Debugger.JustInTime	Just-In-Time debugger	15.0.27005.2	Recommended
Microsoft.VisualStudio.Component.Graphics.Tools	Graphics debugger and GPU profiler for DirectX	15.6.27406.0	Recommended
Microsoft.VisualStudio.Component.Graphics.Win81	Graphics Tools Windows 8.1 SDK	15.6.27406.0	Recommended
Microsoft.VisualStudio.Component.NuGet	NuGet package manager	15.9.28016.0	Recommended
Microsoft.VisualStudio.Component.Static.Analysis.Tools	Static analysis tools	15.0.26208.0	Recommended
Microsoft.VisualStudio.Component.VC.ATL	Visual C++ ATL for x86 and x64	15.7.27625.0	Recommended
Microsoft.VisualStudio.Component.VC.CMake.Project	Visual C++ tools for CMake	15.9.28307.102	Recommended
Microsoft.VisualStudio.Component.VC.DiagnosticTools	C++ profiling tools	15.0.26823.1	Recommended
Microsoft.VisualStudio.Component.VC.TestAdapterForBoostTest	Test Adapter for Boost.Test	15.8.27906.1	Recommended
Microsoft.VisualStudio.Component.VC.TestAdapterForGoogleTest	Test Adapter for Google Test	15.8.27906.1	Recommended
Microsoft.VisualStudio.Component.VC.Tools.x86.x64	VC++ 2017 version 15.9 v14.16 latest v141 tools	15.9.28230.55	Recommended
Microsoft.VisualStudio.Component.Windows10SDK.17763	Windows 10 SDK (10.0.17763.0)	15.9.28307.102	Recommended
Microsoft.VisualStudio.ComponentGroup.WebToolsExtensions	ASP.NET and web development	15.8.27825.0	Recommended
Component.Incredibuild	Incredibuild - Build Acceleration	15.7.27617.1	Optional
Component.IncredibuildMenu	IncredibuildMenu	1.5.0.2	Optional
Microsoft.Component.VC.Runtime.UCRTSDK	Windows Universal CRT SDK	15.6.27309.0	Optional
Microsoft.Net.Component.4.6.1.SDK	.NET Framework 4.6.1 SDK	15.6.27406.0	Optional

Component ID	Name	Version	Dependency Type
Microsoft.Net.Component.4.6.1.TargetingPack	.NET Framework 4.6.1 targeting pack	15.6.27406.0	Optional
Microsoft.VisualStudio.Component.VC.140	VC++ 2015.3 v14.00 (v140) toolset for desktop	15.7.27617.1	Optional
Microsoft.VisualStudio.Component.VC.ATLMFC	Visual C++ MFC for x86 and x64	15.7.27625.0	Optional
Microsoft.VisualStudio.Component.VC.CLI.Support	C++/CLI support	15.6.27309.0	Optional
Microsoft.VisualStudio.Component.VC.Modules.x86.x64	Modules for Standard Library (experimental)	15.6.27309.0	Optional
Microsoft.VisualStudio.Component.Windows10SDK.10240	Windows 10 SDK (10.0.10240.0)	15.6.27406.0	Optional
Microsoft.VisualStudio.Component.Windows10SDK.10586	Windows 10 SDK (10.0.10586.0)	15.6.27406.0	Optional
Microsoft.VisualStudio.Component.Windows10SDK.14393	Windows 10 SDK (10.0.14393.0)	15.6.27406.0	Optional
Microsoft.VisualStudio.Component.Windows10SDK.15063/Desktop	Windows 10 SDK (10.0.15063.0) for Desktop C++ [x86 and x64]	15.6.27406.0	Optional
Microsoft.VisualStudio.Component.Windows10SDK.15063/UWP	Windows 10 SDK (10.0.15063.0) for UWP: C#, VB, JS	15.6.27406.0	Optional
Microsoft.VisualStudio.Component.Windows10SDK.15063/UWP.Native	Windows 10 SDK (10.0.15063.0) for UWP: C++	15.6.27406.0	Optional
Microsoft.VisualStudio.Component.Windows10SDK.16299/Desktop	Windows 10 SDK (10.0.16299.0) for Desktop C++ [x86 and x64]	15.6.27406.0	Optional
Microsoft.VisualStudio.Component.Windows10SDK.16299/Desktop.arm	Windows 10 SDK (10.0.16299.0) for Desktop C++ [ARM and ARM64]	15.6.27406.0	Optional
Microsoft.VisualStudio.Component.Windows10SDK.16299/UWP	Windows 10 SDK (10.0.16299.0) for UWP: C#, VB, JS	15.6.27406.0	Optional
Microsoft.VisualStudio.Component.Windows10SDK.16299/UWP.Native	Windows 10 SDK (10.0.16299.0) for UWP: C++	15.6.27406.0	Optional

Component ID	Name	Version	Dependency Type
Microsoft.VisualStudio.Component.Windows10SDK.17134	Windows 10 SDK (10.0.17134.0)	15.9.28307.102	Optional
Microsoft.VisualStudio.Component.Windows81SDK	Windows 8.1 SDK	15.6.27406.0	Optional
Microsoft.VisualStudio.Component.WinXP	Windows XP support for C++	15.8.27924.0	Optional
Microsoft.VisualStudio.ComponentGroup.NativeDesktop.Win81	Windows 8.1 SDK and UCRT SDK	15.6.27406.0	Optional
Microsoft.VisualStudio.ComponentGroup.NativeDesktop.WinXP	Windows XP support for C++	15.8.27705.0	Optional
Microsoft.VisualStudio.ComponentGroup.Windows10SDK.15063	Windows 10 SDK (10.0.15063.0)	15.8.27825.0	Optional
Microsoft.VisualStudio.ComponentGroup.Windows10SDK.16299	Windows 10 SDK (10.0.16299.0)	15.8.27825.0	Optional

Game development with C++

ID: Microsoft.VisualStudio.Workload.NativeGame

Description: Use the full power of C++ to build professional games powered by DirectX, Unreal, or Cocos2d.

Components included by this workload

Component ID	Name	Version	Dependency Type
Microsoft.VisualStudio.Component.Static.Analysis.Tools	Static analysis tools	15.0.26208.0	Required
Microsoft.VisualStudio.Component.VC.Corelde	Visual Studio C++ core features	15.6.27406.0	Required
Microsoft.VisualStudio.Component.VC.Redist.14.Latest	Visual C++ 2017 Redistributable Update	15.6.27406.0	Required
Microsoft.VisualStudio.Component.VC.Tools.x86.x64	VC++ 2017 version 15.9 v14.16 latest v141 tools	15.9.28230.55	Required
Microsoft.VisualStudio.Component.Windows10SDK	Windows Universal C Runtime	15.6.27406.0	Required
Microsoft.VisualStudio.Component.Graphics.Tools	Graphics debugger and GPU profiler for DirectX	15.6.27406.0	Recommended

Component ID	Name	Version	Dependency Type
Microsoft.VisualStudio.Component.Graphics.Win81	Graphics Tools Windows 8.1 SDK	15.6.27406.0	Recommended
Microsoft.VisualStudio.Component.VC.DiagnosticTools	C++ profiling tools	15.0.26823.1	Recommended
Microsoft.VisualStudio.Component.Windows10SDK.17763	Windows 10 SDK (10.0.17763.0)	15.9.28307.102	Recommended
Component.Android.NDK.R12B	Android NDK (R12B)	12.1.10	Optional
Component.Android.SDK23.Private	Android SDK setup (API level 23) (local install for Mobile development with JavaScript / C++)	15.9.28016.0	Optional
Component.Ant	Apache Ant (1.9.3)	1.9.3.8	Optional
Component.Cocos	Cocos	15.0.26906.1	Optional
Component.Incredibuild	Incredibuild - Build Acceleration	15.7.27617.1	Optional
Component.IncredibuildMenu	IncredibuildMenu	1.5.0.2	Optional
Component.MDD.Android	C++ Android development tools	15.0.26606.0	Optional
Component.OpenJDK	Microsoft distribution OpenJDK	15.9.28125.51	Optional
Component.Unreal	Unreal Engine installer	15.8.27729.1	Optional
Component.Unreal.Android	Visual Studio Android support for Unreal Engine	15.9.28307.341	Optional
Microsoft.Component.VC_Runtime.UCRTSDK	Windows Universal CRT SDK	15.6.27309.0	Optional
Microsoft.Net.Component.4.5.1.TargetingPack	.NET Framework 4.5.1 targeting pack	15.6.27406.0	Optional
Microsoft.Net.Component.4.5.2.TargetingPack	.NET Framework 4.5.2 targeting pack	15.6.27406.0	Optional
Microsoft.Net.Component.4.5.TargetingPack	.NET Framework 4.5 targeting pack	15.6.27406.0	Optional
Microsoft.Net.Component.4.6.1.SDK	.NET Framework 4.6.1 SDK	15.6.27406.0	Optional

Component ID	Name	Version	Dependency Type
Microsoft.Net.Component.4.6.1.TargetingPack	.NET Framework 4.6.1 targeting pack	15.6.27406.0	Optional
Microsoft.Net.Component.4.6.TargetingPack	.NET Framework 4.6 targeting pack	15.6.27406.0	Optional
Microsoft.Net.Component.4.TargetingPack	.NET Framework 4 targeting pack	15.6.27406.0	Optional
Microsoft.Net.ComponentGroup.DevelopmentPrerequisites	.NET Framework 4.6.1 development tools	15.8.27825.0	Optional
Microsoft.Net.ComponentGroup.TargetingPacks.Common	.NET Framework 4 – 4.6 development tools	15.6.27406.0	Optional
Microsoft.VisualStudio.Component.NuGet.BuildTools	NuGet targets and build tasks	15.9.28016.0	Optional
Microsoft.VisualStudio.Component.Roslyn.Compiler	C# and Visual Basic Roslyn compilers	15.6.27309.0	Optional
Microsoft.VisualStudio.Component.Roslyn.LanguageServices	C# and Visual Basic	15.8.27729.1	Optional
Microsoft.VisualStudio.Component.Windows10SDK.10240	Windows 10 SDK (10.0.10240.0)	15.6.27406.0	Optional
Microsoft.VisualStudio.Component.Windows10SDK.10586	Windows 10 SDK (10.0.10586.0)	15.6.27406.0	Optional
Microsoft.VisualStudio.Component.Windows10SDK.14393	Windows 10 SDK (10.0.14393.0)	15.6.27406.0	Optional
Microsoft.VisualStudio.Component.Windows10SDK.15063.Desktop	Windows 10 SDK (10.0.15063.0) for Desktop C++ [x86 and x64]	15.6.27406.0	Optional
Microsoft.VisualStudio.Component.Windows10SDK.15063.UWP	Windows 10 SDK (10.0.15063.0) for UWP: C#, VB, JS	15.6.27406.0	Optional
Microsoft.VisualStudio.Component.Windows10SDK.15063.UWP.Native	Windows 10 SDK (10.0.15063.0) for UWP: C++	15.6.27406.0	Optional
Microsoft.VisualStudio.Component.Windows10SDK.16299.Desktop	Windows 10 SDK (10.0.16299.0) for Desktop C++ [x86 and x64]	15.6.27406.0	Optional

Component ID	Name	Version	Dependency Type
Microsoft.VisualStudio.Component.Windows10SDK.16299.9.Desktop.arm	Windows 10 SDK (10.0.16299.0) for Desktop C++ [ARM and ARM64]	15.6.27406.0	Optional
Microsoft.VisualStudio.Component.Windows10SDK.16299.UWP	Windows 10 SDK (10.0.16299.0) for UWP: C#, VB, JS	15.6.27406.0	Optional
Microsoft.VisualStudio.Component.Windows10SDK.16299.UWP.Native	Windows 10 SDK (10.0.16299.0) for UWP: C++	15.6.27406.0	Optional
Microsoft.VisualStudio.Component.Windows10SDK.17134	Windows 10 SDK (10.0.17134.0)	15.9.28307.102	Optional
Microsoft.VisualStudio.Component.Windows81SDK	Windows 8.1 SDK	15.6.27406.0	Optional
Microsoft.VisualStudio.ComponentGroup.NativeDesktop.Win81	Windows 8.1 SDK and UCRT SDK	15.6.27406.0	Optional
Microsoft.VisualStudio.ComponentGroup.Windows10SDK.15063	Windows 10 SDK (10.0.15063.0)	15.8.27825.0	Optional
Microsoft.VisualStudio.ComponentGroup.Windows10SDK.16299	Windows 10 SDK (10.0.16299.0)	15.8.27825.0	Optional

Mobile development with C++

ID: Microsoft.VisualStudio.Workload.NativeMobile

Description: Build cross-platform applications for iOS, Android or Windows using C++.

Components included by this workload

Component ID	Name	Version	Dependency Type
Component.Android.SDK19.Private	Android SDK setup (API level 19) (local install for Mobile development with JavaScript / C++)	15.9.28107.0	Required
Component.Android.SDK21.Private	Android SDK setup (API level 21) (local install for Mobile development with JavaScript / C++)	15.9.28016.0	Required
Component.Android.SDK22.Private	Android SDK setup (API level 22) (local install for Mobile development with JavaScript / C++)	15.9.28016.0	Required

Component ID	Name	Version	Dependency Type
Component.Android.SDK23.Private	Android SDK setup (API level 23) (local install for Mobile development with JavaScript / C++)	15.9.28016.0	Required
Component.Android.SDK25.Private	Android SDK setup (API level 25) (local install for Mobile development with JavaScript / C++)	15.9.28016.0	Required
Component.OpenJDK	Microsoft distribution OpenJDK	15.9.28125.51	Required
Microsoft.VisualStudio.Component.VC.Corede	Visual Studio C++ core features	15.6.27406.0	Required
Component.Android.NDK.R15C	Android NDK (R15C)	15.2.1	Recommended
Component.Ant	Apache Ant (1.9.3)	1.9.3.8	Recommended
Component.MDD.Android	C++ Android development tools	15.0.26606.0	Recommended
Component.Android.NDK.R12B	Android NDK (R12B)	12.1.10	Optional
Component.Android.NDK.R12B_3264	Android NDK (R12B) (32bit)	12.1.11	Optional
Component.Android.NDK.R13B	Android NDK (R13B)	13.1.7	Optional
Component.Android.NDK.R13B_3264	Android NDK (R13B) (32bit)	13.1.8	Optional
Component.Android.NDK.R15C_3264	Android NDK (R15C) (32bit)	15.2.1	Optional
Component.Google.Android.Emulator.API23.Private	Google Android Emulator (API Level 23) (local install)	15.6.27413.0	Optional
Component.HAXM.Private	Intel Hardware Accelerated Execution Manager (HAXM) (local install)	15.9.28307.421	Optional
Component.Incredibuild	Incredibuild - Build Acceleration	15.7.27617.1	Optional
Component.IncredibuildMenu	IncredibuildMenu	1.5.0.2	Optional
Component.MDD.IOS	C++ iOS development tools	15.0.26621.2	Optional

.NET Core cross-platform development

ID: Microsoft.VisualStudio.Workload.NetCoreTools

Description: Build cross-platform applications using .NET Core, ASP.NET Core, HTML/JavaScript, and Containers including Docker support.

Components included by this workload

Component ID	Name	Version	Dependency Type
Component.Microsoft.VisualStudio.RazorExtension	Razor Language Services	15.0.26720.2	Required
Component.Microsoft.Web.LibraryManager	Library Manager	15.8.27705.0	Required
Component.WebSocket	WebSocket4Net	15.0.26606.0	Required
Microsoft.Component.ClickOnce	ClickOnce Publishing	15.8.27825.0	Required
Microsoft.Component.MSBuild	MSBuild	15.7.27520.0	Required
Microsoft.Net.Component.4.5.2.TargetingPack	.NET Framework 4.5.2 targeting pack	15.6.27406.0	Required
Microsoft.Net.Component.4.5.TargetingPack	.NET Framework 4.5 targeting pack	15.6.27406.0	Required
Microsoft.Net.Component.4.6.1.SDK	.NET Framework 4.6.1 SDK	15.6.27406.0	Required
Microsoft.Net.Component.4.6.1.TargetingPack	.NET Framework 4.6.1 targeting pack	15.6.27406.0	Required
Microsoft.Net.ComponentGroup.DevelopmentPrerequisites	.NET Framework 4.6.1 development tools	15.8.27825.0	Required
Microsoft.Net.Core.Component.SDK.2.1	.NET Core 2.1 development tools	15.8.27924.0	Required
Microsoft.NetCore.ComponentGroup.DevelopmentTools.2.1	.NET Core 2.1 development tools	15.8.27924.0	Required
Microsoft.NetCore.ComponentGroup.Web.2.1	.NET Core 2.1 development tools	15.8.27924.0	Required
Microsoft.VisualStudio.Component.Common.Azure.Tools	Connectivity and publishing tools	15.9.28107.0	Required
Microsoft.VisualStudio.Component.DockerTools	Container development tools	15.8.27906.1	Required

Component ID	Name	Version	Dependency Type
Microsoft.VisualStudio.Component.DockerTools.BuildTools	Container development tools - Build Tools	15.7.27617.1	Required
Microsoft.VisualStudio.Component.FSharp	F# language support	15.8.27825.0	Required
Microsoft.VisualStudio.Component.FSharp.WebTemplates	F# language support for web projects	15.9.28307.421	Required
Microsoft.VisualStudio.Component.IIExpress	IIS Express	15.0.26208.0	Required
Microsoft.VisualStudio.Component.JavaScript.Diagnostics	JavaScript diagnostics	15.8.27729.1	Required
Microsoft.VisualStudio.Component.JavaScript.TypeScript	JavaScript and TypeScript language support	15.9.28125.51	Required
Microsoft.VisualStudio.Component.ManagedDesktop.Core	Managed Desktop Workload Core	15.8.27729.1	Required
Microsoft.VisualStudio.Component.NuGet	NuGet package manager	15.9.28016.0	Required
Microsoft.VisualStudio.Component.PortableLibrary	.NET Portable Library targeting pack	15.6.27309.0	Required
Microsoft.VisualStudio.Component.Roslyn.Compiler	C# and Visual Basic Roslyn compilers	15.6.27309.0	Required
Microsoft.VisualStudio.Component.Roslyn.LanguageServices	C# and Visual Basic	15.8.27729.1	Required
Microsoft.VisualStudio.Component.SQL.ADAL	SQL ADAL runtime	15.6.27406.0	Required
Microsoft.VisualStudio.Component.SQL.CLR	CLR data types for SQL Server	15.0.26208.0	Required
Microsoft.VisualStudio.Component.SQL.CMDUtils	SQL Server Command Line Utilities	15.0.26208.0	Required
Microsoft.VisualStudio.Component.SQL.DataSources	Data sources for SQL Server support	15.0.26621.2	Required
Microsoft.VisualStudio.Component.SQL.LocalDB.Runtime	SQL Server Express 2016 LocalDB	15.7.27617.1	Required
Microsoft.VisualStudio.Component.SQL.NCLI	SQL Server Native Client	15.0.26208.0	Required

Component ID	Name	Version	Dependency Type
Microsoft.VisualStudio.Component.SQL.SSDT	SQL Server Data Tools	15.9.28107.0	Required
Microsoft.VisualStudio.Component.Static.Analysis.Tools	Static analysis tools	15.0.26208.0	Required
Microsoft.VisualStudio.Component.TextTemplating	Text Template Transformation	15.0.26208.0	Required
Microsoft.VisualStudio.Component.TypeScript.3.1	TypeScript 3.1 SDK	15.0.28218.60	Required
Microsoft.VisualStudio.Component.VisualStudioData	Data sources and service references	15.6.27406.0	Required
Microsoft.VisualStudio.ComponentGroup.Web	ASP.NET and web development tools prerequisites	15.9.28219.51	Required
Microsoft.VisualStudio.ComponentGroup.WebToolsExtensions	ASP.NET and web development	15.8.27825.0	Required
Component.Microsoft.VisualStudio.Web.AzureFunctions	Microsoft Azure WebJobs Tools	15.7.27617.1	Recommended
Microsoft.VisualStudio.Component.AppInsights.Tools	Developer Analytics tools	15.8.27825.0	Recommended
Microsoft.VisualStudio.Component.Azure.AuthoringTools	Azure Authoring Tools	15.9.28307.421	Recommended
Microsoft.VisualStudio.Component.Azure.ClientLibs	Azure libraries for .NET	15.0.26208.0	Recommended
Microsoft.VisualStudio.Component.Azure.Compute.Emulator	Azure Compute Emulator	15.9.28307.421	Recommended
Microsoft.VisualStudio.Component.Azure.Storage.Emulator	Azure Storage Emulator	15.9.28125.51	Recommended
Microsoft.VisualStudio.Component.CloudExplorer	Cloud Explorer	15.9.28230.55	Recommended
Microsoft.VisualStudio.Component.DiagnosticTools	.NET profiling tools	15.8.27729.1	Recommended
Microsoft.VisualStudio.Component.Web	ASP.NET and web development tools	15.8.27825.0	Recommended
Microsoft.VisualStudio.Component.WebDeploy	Web Deploy	15.8.27729.1	Recommended

COMPONENT ID	NAME	VERSION	DEPENDENCY TYPE
Microsoft.VisualStudio.ComponentGroup.AzureFunctions	Microsoft Azure WebJobs Tools	15.7.27617.1	Recommended
Microsoft.VisualStudio.ComponentGroup.Web.CloudTools	Cloud tools for web development	15.8.27729.1	Recommended
Microsoft.Net.Core.Component.SDK	.NET Core 2.0 development tools	15.6.27406.0	Optional
Microsoft.Net.Core.Component.SDK.1x	.NET Core 1.0 - 1.1 development tools	15.6.27406.0	Optional
Microsoft.NetCore.1x.ComponentGroup.Web	.NET Core 1.0 - 1.1 development tools for Web	15.6.27406.0	Optional
Microsoft.NetCore.ComponentGroup.DevelopmentTools	.NET Core 2.0 development tools	15.8.27729.1	Optional
Microsoft.NetCore.ComponentGroup.Web	.NET Core 2.0 development tools	15.7.27625.0	Optional
Microsoft.VisualStudio.ComponentGroup.IISDevelopment	Development time IIS support	15.9.28219.51	Optional

Mobile development with .NET

ID: Microsoft.VisualStudio.Workload.NetCrossPlat

Description: Build cross-platform applications for iOS, Android or Windows using Xamarin.

Components included by this workload

COMPONENT ID	NAME	VERSION	DEPENDENCY TYPE
Component.Xamarin	Xamarin	15.8.27906.1	Required
Component.Xamarin.RemotedSimulator	Xamarin Remoted Simulator	15.6.27323.2	Required
Microsoft.Component.MSBuild	MSBuild	15.7.27520.0	Required
Microsoft.Net.Component.4.6.1.SDK	.NET Framework 4.6.1 SDK	15.6.27406.0	Required
Microsoft.Net.Component.4.6.1.TargetingPack	.NET Framework 4.6.1 targeting pack	15.6.27406.0	Required
Microsoft.Net.ComponentGroup.DevelopmentPrerequisites	.NET Framework 4.6.1 development tools	15.8.27825.0	Required
Microsoft.Net.Core.Component.SDK	.NET Core 2.0 development tools	15.6.27406.0	Required

Component ID	Name	Version	Dependency Type
Microsoft.NetCore.ComponentGroup.DevelopmentTools	.NET Core 2.0 development tools	15.8.27729.1	Required
Microsoft.VisualStudio.Component.FSharp	F# language support	15.8.27825.0	Required
Microsoft.VisualStudio.Component.Merq	Common Xamarin internal tools	15.8.27924.0	Required
Microsoft.VisualStudio.Component.MonoDebugger	Mono debugger	15.0.26720.2	Required
Microsoft.VisualStudio.Component.NuGet	NuGet package manager	15.9.28016.0	Required
Microsoft.VisualStudio.Component.PortableLibrary	.NET Portable Library targeting pack	15.6.27309.0	Required
Microsoft.VisualStudio.Component.Roslyn.Compiler	C# and Visual Basic Roslyn compilers	15.6.27309.0	Required
Microsoft.VisualStudio.Component.Roslyn.LanguageServices	C# and Visual Basic	15.8.27729.1	Required
Microsoft.VisualStudio.Component.Static.Analysis.Tools	Static analysis tools	15.0.26208.0	Required
Microsoft.VisualStudio.ComponentGroup.WebToolsExtensions.TemplateEngine	ASP.NET templating engine	15.8.27729.1	Required
Component.Android.SDK27	Android SDK setup (API level 27)	15.9.28016.0	Recommended
Component.Google.Android.Emulator.API27	Google Android Emulator (API Level 27)	15.9.28307.421	Recommended
Component.HAXM	Intel Hardware Accelerated Execution Manager (HAXM) (global install)	15.9.28307.421	Recommended
Component.OpenJDK	Microsoft distribution OpenJDK	15.9.28125.51	Recommended
Component.Xamarin.Inspect	Xamarin Workbooks	15.0.26606.0	Optional
Microsoft.Component.ClickOnce	ClickOnce Publishing	15.8.27825.0	Optional
Microsoft.Component.NetFX.Native	.NET Native	15.0.26208.0	Optional

Component ID	Name	Version	Dependency Type
Microsoft.VisualStudio.Component.AppInsights.Tools	Developer Analytics tools	15.8.27825.0	Optional
Microsoft.VisualStudio.Component.DiagnosticTools	.NET profiling tools	15.8.27729.1	Optional
Microsoft.VisualStudio.Component.Graphics	Image and 3D model editors	15.6.27406.0	Optional
Microsoft.VisualStudio.Component.SQL.CLR	CLR data types for SQL Server	15.0.26208.0	Optional
Microsoft.VisualStudio.Component.VisualStudioData	Data sources and service references	15.6.27406.0	Optional
Microsoft.VisualStudio.Component.Windows10SDK.17763	Windows 10 SDK (10.0.17763.0)	15.9.28307.102	Optional
Microsoft.VisualStudio.ComponentGroup.UWP.Xamarin	Universal Windows Platform tools for Xamarin	15.9.28307.102	Optional

ASP.NET and web development

ID: Microsoft.VisualStudio.Workload.NetWeb

Description: Build web applications using ASP.NET, ASP.NET Core, HTML/JavaScript, and Containers including Docker support.

Components included by this workload

Component ID	Name	Version	Dependency Type
Component.Microsoft.VisualStudio.RazorExtension	Razor Language Services	15.0.26720.2	Required
Component.Microsoft.Web.LibraryManager	Library Manager	15.8.27705.0	Required
Component.WebSocket	WebSocket4Net	15.0.26606.0	Required
Microsoft.Component.ClickOnce	ClickOnce Publishing	15.8.27825.0	Required
Microsoft.Component.MSBuild	MSBuild	15.7.27520.0	Required
Microsoft.Net.Component.4.5.2.TargetingPack	.NET Framework 4.5.2 targeting pack	15.6.27406.0	Required
Microsoft.Net.Component.4.5.TargetingPack	.NET Framework 4.5 targeting pack	15.6.27406.0	Required

Component ID	Name	Version	Dependency Type
Microsoft.Net.Component.4.6.1.SDK	.NET Framework 4.6.1 SDK	15.6.27406.0	Required
Microsoft.Net.Component.4.6.1.TargetingPack	.NET Framework 4.6.1 targeting pack	15.6.27406.0	Required
Microsoft.Net.ComponentGroup.DevelopmentPrerequisites	.NET Framework 4.6.1 development tools	15.8.27825.0	Required
Microsoft.Net.Core.Component.SDK.2.1	.NET Core 2.1 development tools	15.8.27924.0	Required
Microsoft.NetCore.ComponentGroup.DevelopmentTools.2.1	.NET Core 2.1 development tools	15.8.27924.0	Required
Microsoft.NetCore.ComponentGroup.Web.2.1	.NET Core 2.1 development tools	15.8.27924.0	Required
Microsoft.VisualStudio.Component.Common.Azure.Tools	Connectivity and publishing tools	15.9.28107.0	Required
Microsoft.VisualStudio.Component.DockerTools	Container development tools	15.8.27906.1	Required
Microsoft.VisualStudio.Component.DockerTools.BuildTools	Container development tools - Build Tools	15.7.27617.1	Required
Microsoft.VisualStudio.Component.FSharp	F# language support	15.8.27825.0	Required
Microsoft.VisualStudio.Component.FSharp.WebTemplates	F# language support for web projects	15.9.28307.421	Required
Microsoft.VisualStudio.Component.IISExpress	IIS Express	15.0.26208.0	Required
Microsoft.VisualStudio.Component.JavaScript.Diagnostics	JavaScript diagnostics	15.8.27729.1	Required
Microsoft.VisualStudio.Component.JavaScript.TypeScript	JavaScript and TypeScript language support	15.9.28125.51	Required
Microsoft.VisualStudio.Component.ManagedDesktop.Core	Managed Desktop Workload Core	15.8.27729.1	Required
Microsoft.VisualStudio.Component.NuGet	NuGet package manager	15.9.28016.0	Required
Microsoft.VisualStudio.Component.PortableLibrary	.NET Portable Library targeting pack	15.6.27309.0	Required

Component ID	Name	Version	Dependency Type
Microsoft.VisualStudio.Component.Roslyn.Compiler	C# and Visual Basic Roslyn compilers	15.6.27309.0	Required
Microsoft.VisualStudio.Component.Roslyn.LanguageServices	C# and Visual Basic	15.8.27729.1	Required
Microsoft.VisualStudio.Component.SQL.ADAL	SQL ADAL runtime	15.6.27406.0	Required
Microsoft.VisualStudio.Component.SQL.CLR	CLR data types for SQL Server	15.0.26208.0	Required
Microsoft.VisualStudio.Component.SQL.CMDUtils	SQL Server Command Line Utilities	15.0.26208.0	Required
Microsoft.VisualStudio.Component.SQL.DataSources	Data sources for SQL Server support	15.0.26621.2	Required
Microsoft.VisualStudio.Component.SQL.LocalDB.Runtime	SQL Server Express 2016 LocalDB	15.7.27617.1	Required
Microsoft.VisualStudio.Component.SQL.NCLI	SQL Server Native Client	15.0.26208.0	Required
Microsoft.VisualStudio.Component.SQL.SSDT	SQL Server Data Tools	15.9.28107.0	Required
Microsoft.VisualStudio.Component.Static.Analysis.Tools	Static analysis tools	15.0.26208.0	Required
Microsoft.VisualStudio.Component.TextTemplating	Text Template Transformation	15.0.26208.0	Required
Microsoft.VisualStudio.Component.TypeScript.3.1	TypeScript 3.1 SDK	15.0.28218.60	Required
Microsoft.VisualStudio.Component.VisualStudioData	Data sources and service references	15.6.27406.0	Required
Microsoft.VisualStudio.Component.Web	ASP.NET and web development tools	15.8.27825.0	Required
Microsoft.VisualStudio.ComponentGroup.Web	ASP.NET and web development tools prerequisites	15.9.28219.51	Required
Microsoft.VisualStudio.ComponentGroup.WebToolsExtensions	ASP.NET and web development	15.8.27825.0	Required
Component.Microsoft.VisualStudio.Web.AzureFunctions	Microsoft Azure WebJobs Tools	15.7.27617.1	Recommended

Component ID	Name	Version	Dependency Type
Microsoft.Net.Component.4.5.1.TargetingPack	.NET Framework 4.5.1 targeting pack	15.6.27406.0	Recommended
Microsoft.Net.Component.4.6.TargetingPack	.NET Framework 4.6 targeting pack	15.6.27406.0	Recommended
Microsoft.Net.Component.4.TargetingPack	.NET Framework 4 targeting pack	15.6.27406.0	Recommended
Microsoft.Net.ComponentGroup.TargetingPacks.Common	.NET Framework 4 – 4.6 development tools	15.6.27406.0	Recommended
Microsoft.VisualStudio.Component.AppInsights.Tools	Developer Analytics tools	15.8.27825.0	Recommended
Microsoft.VisualStudio.Component.AspNet45	Advanced ASP.NET features	15.7.27625.0	Recommended
Microsoft.VisualStudio.Component.Azure.AuthoringTools	Azure Authoring Tools	15.9.28307.421	Recommended
Microsoft.VisualStudio.Component.Azure.ClientLibs	Azure libraries for .NET	15.0.26208.0	Recommended
Microsoft.VisualStudio.Component.Azure.Compute.Emulator	Azure Compute Emulator	15.9.28307.421	Recommended
Microsoft.VisualStudio.Component.Azure.Storage.Emulator	Azure Storage Emulator	15.9.28125.51	Recommended
Microsoft.VisualStudio.Component.CloudExplorer	Cloud Explorer	15.9.28230.55	Recommended
Microsoft.VisualStudio.Component.DiagnosticTools	.NET profiling tools	15.8.27729.1	Recommended
Microsoft.VisualStudio.Component.EntityFramework	Entity Framework 6 tools	15.6.27406.0	Recommended
Microsoft.VisualStudio.Component.Wcf.Tooling	Windows Communication Foundation	15.8.27924.0	Recommended
Microsoft.VisualStudio.Component.WebDeploy	Web Deploy	15.8.27729.1	Recommended
Microsoft.VisualStudio.ComponentGroup.AzureFunctions	Microsoft Azure WebJobs Tools	15.7.27617.1	Recommended
Microsoft.VisualStudio.ComponentGroup.Web.CloudTools	Cloud tools for web development	15.8.27729.1	Recommended

Component ID	Name	Version	Dependency Type
Microsoft.Net.Component.4.6.2.SDK	.NET Framework 4.6.2 SDK	15.6.27406.0	Optional
Microsoft.Net.Component.4.6.2.TargetingPack	.NET Framework 4.6.2 targeting pack	15.6.27406.0	Optional
Microsoft.Net.Component.4.7.1.SDK	.NET Framework 4.7.1 SDK	15.6.27406.0	Optional
Microsoft.Net.Component.4.7.1.TargetingPack	.NET Framework 4.7.1 targeting pack	15.6.27406.0	Optional
Microsoft.Net.Component.4.7.2.SDK	.NET Framework 4.7.2 SDK	15.8.27825.0	Optional
Microsoft.Net.Component.4.7.2.TargetingPack	.NET Framework 4.7.2 targeting pack	15.8.27825.0	Optional
Microsoft.Net.Component.4.7.SDK	.NET Framework 4.7 SDK	15.6.27406.0	Optional
Microsoft.Net.Component.4.7.TargetingPack	.NET Framework 4.7 targeting pack	15.6.27406.0	Optional
Microsoft.Net.ComponentGroup.4.6.2.DeveloperTools	.NET Framework 4.6.2 development tools	15.6.27406.0	Optional
Microsoft.Net.ComponentGroup.4.7.1.DeveloperTools	.NET Framework 4.7.1 development tools	15.6.27406.0	Optional
Microsoft.Net.ComponentGroup.4.7.2.DeveloperTools	.NET Framework 4.7.2 development tools	15.8.27825.0	Optional
Microsoft.Net.ComponentGroup.4.7.DeveloperTools	.NET Framework 4.7 development tools	15.6.27406.0	Optional
Microsoft.Net.Core.Component.SDK	.NET Core 2.0 development tools	15.6.27406.0	Optional
Microsoft.Net.Core.Component.SDK.1x	.NET Core 1.0 - 1.1 development tools	15.6.27406.0	Optional
Microsoft.NetCore.1x.ComponentGroup.Web	.NET Core 1.0 - 1.1 development tools for Web	15.6.27406.0	Optional
Microsoft.NetCore.ComponentGroup.DevelopmentTools	.NET Core 2.0 development tools	15.8.27729.1	Optional
Microsoft.NetCore.ComponentGroup.Web	.NET Core 2.0 development tools	15.7.27625.0	Optional
Microsoft.VisualStudio.ComponentGroup.IISDevelopment	Development time IIS support	15.9.28219.51	Optional

Component ID	Name	Version	Dependency Type
Microsoft.VisualStudio.Web.Mvc4.ComponentGroup	ASP.NET MVC 4	15.6.27406.0	Optional

Node.js development

ID: Microsoft.VisualStudio.Workload.Node

Description: Build scalable network applications using Node.js, an asynchronous event-driven JavaScript runtime.

Components included by this workload

Component ID	Name	Version	Dependency Type
Component.WebSocket	WebSocket4Net	15.0.26606.0	Required
Microsoft.VisualStudio.Component.JavaScript.Diagnostics	JavaScript diagnostics	15.8.27729.1	Required
Microsoft.VisualStudio.Component.JavaScript.TypeScript	JavaScript and TypeScript language support	15.9.28125.51	Required
Microsoft.VisualStudio.Component.Node.Build	Node.js MSBuild support	15.8.27825.0	Required
Microsoft.VisualStudio.Component.Node.Tools	Node.js development support	15.8.27825.0	Required
Microsoft.VisualStudio.Component.NuGet	NuGet package manager	15.9.28016.0	Required
Microsoft.VisualStudio.Component.TestTools.Core	Testing tools core features	15.7.27520.0	Required
Microsoft.VisualStudio.Component.TypeScript.3.1	TypeScript 3.1 SDK	15.0.28218.60	Required
Microsoft.VisualStudio.ComponentGroup.WebToolsExtensions	ASP.NET and web development	15.8.27825.0	Required
Microsoft.VisualStudio.Component.WebDeploy	Web Deploy	15.8.27729.1	Recommended
Microsoft.VisualStudio.Component.AppInsights.Tools	Developer Analytics tools	15.8.27825.0	Optional
Microsoft.VisualStudio.Component.Common.Azure.Tools	Connectivity and publishing tools	15.9.28107.0	Optional
Microsoft.VisualStudio.Component.Static.Analysis.Tools	Static analysis tools	15.0.26208.0	Optional

Component ID	Name	Version	Dependency Type
Microsoft.VisualStudio.Component.VC.CoreRedist	Visual Studio C++ core features	15.6.27406.0	Optional
Microsoft.VisualStudio.Component.VC.Tools.x86.x64	VC++ 2017 version 15.9 v14.16 latest v141 tools	15.9.28230.55	Optional

Office/SharePoint development

ID: Microsoft.VisualStudio.Workload.Office

Description: Create Office and SharePoint add-ins, SharePoint solutions, and VSTO add-ins using C#, VB, and JavaScript.

Components included by this workload

Component ID	Name	Version	Dependency Type
Component.Microsoft.VisualStudio.RazorExtension	Razor Language Services	15.0.26720.2	Required
Component.Microsoft.Web.LibraryManager	Library Manager	15.8.27705.0	Required
Component.WebSocket	WebSocket4Net	15.0.26606.0	Required
Microsoft.Component.ClickOnce	ClickOnce Publishing	15.8.27825.0	Required
Microsoft.Component.MSBuild	MSBuild	15.7.27520.0	Required
Microsoft.Net.Component.4.5.2.TargetingPack	.NET Framework 4.5.2 targeting pack	15.6.27406.0	Required
Microsoft.Net.Component.4.5.TargetingPack	.NET Framework 4.5 targeting pack	15.6.27406.0	Required
Microsoft.Net.Component.4.6.1.SDK	.NET Framework 4.6.1 SDK	15.6.27406.0	Required
Microsoft.Net.Component.4.6.1.TargetingPack	.NET Framework 4.6.1 targeting pack	15.6.27406.0	Required
Microsoft.Net.Component.4.TargetingPack	.NET Framework 4 targeting pack	15.6.27406.0	Required
Microsoft.Net.ComponentGroup.DevelopmentPrerequisites	.NET Framework 4.6.1 development tools	15.8.27825.0	Required
Microsoft.Net.Core.Component.SDK.2.1	.NET Core 2.1 development tools	15.8.27924.0	Required

Component ID	Name	Version	Dependency Type
Microsoft.VisualStudio.Component.AppInsights.Tools	Developer Analytics tools	15.8.27825.0	Required
Microsoft.VisualStudio.Component.Common.Azure.Tools	Connectivity and publishing tools	15.9.28107.0	Required
Microsoft.VisualStudio.Component.DockerTools	Container development tools	15.8.27906.1	Required
Microsoft.VisualStudio.Component.DockerTools.BuildTools	Container development tools - Build Tools	15.7.27617.1	Required
Microsoft.VisualStudio.Component.IISExpress	IIS Express	15.0.26208.0	Required
Microsoft.VisualStudio.Component.JavaScript.Diagnostics	JavaScript diagnostics	15.8.27729.1	Required
Microsoft.VisualStudio.Component.JavaScript.TypeScript	JavaScript and TypeScript language support	15.9.28125.51	Required
Microsoft.VisualStudio.Component.ManagedDesktop.Core	Managed Desktop Workload Core	15.8.27729.1	Required
Microsoft.VisualStudio.Component.ManagedDesktop.Prerequisites	.NET desktop development tools	15.7.27625.0	Required
Microsoft.VisualStudio.Component.NuGet	NuGet package manager	15.9.28016.0	Required
Microsoft.VisualStudio.Component.PortableLibrary	.NET Portable Library targeting pack	15.6.27309.0	Required
Microsoft.VisualStudio.Component.Roslyn.Compiler	C# and Visual Basic Roslyn compilers	15.6.27309.0	Required
Microsoft.VisualStudio.Component.Roslyn.LanguageServices	C# and Visual Basic	15.8.27729.1	Required
Microsoft.VisualStudio.Component.Sharepoint.Tools	Office Developer Tools for Visual Studio	15.8.27924.0	Required
Microsoft.VisualStudio.Component.SQL.ADAL	SQL ADAL runtime	15.6.27406.0	Required
Microsoft.VisualStudio.Component.SQL.CLR	CLR data types for SQL Server	15.0.26208.0	Required
Microsoft.VisualStudio.Component.SQL.CMDUtils	SQL Server Command Line Utilities	15.0.26208.0	Required

Component ID	Name	Version	Dependency Type
Microsoft.VisualStudio.Component.SQL.DataSources	Data sources for SQL Server support	15.0.26621.2	Required
Microsoft.VisualStudio.Component.SQL.LocalDB.Runtime	SQL Server Express 2016 LocalDB	15.7.27617.1	Required
Microsoft.VisualStudio.Component.SQL.NCLI	SQL Server Native Client	15.0.26208.0	Required
Microsoft.VisualStudio.Component.SQL.SSDT	SQL Server Data Tools	15.9.28107.0	Required
Microsoft.VisualStudio.Component.Static.Analysis.Tools	Static analysis tools	15.0.26208.0	Required
Microsoft.VisualStudio.Component.TextTemplating	Text Template Transformation	15.0.26208.0	Required
Microsoft.VisualStudio.Component.TypeScript.3.1	TypeScript 3.1 SDK	15.0.28218.60	Required
Microsoft.VisualStudio.Component.VisualStudioData	Data sources and service references	15.6.27406.0	Required
Microsoft.VisualStudio.Component.Wcf.Tooling	Windows Communication Foundation	15.8.27924.0	Required
Microsoft.VisualStudio.Component.Web	ASP.NET and web development tools	15.8.27825.0	Required
Microsoft.VisualStudio.Component.Workflow	Windows Workflow Foundation	15.8.27825.0	Required
Microsoft.VisualStudio.ComponentGroup.Web	ASP.NET and web development tools prerequisites	15.9.28219.51	Required
Microsoft.VisualStudio.ComponentGroup.WebToolsExtensions	ASP.NET and web development	15.8.27825.0	Required
Microsoft.VisualStudio.Component.TeamOffice	Visual Studio Tools for Office (VSTO)	15.7.27625.0	Recommended
Microsoft.VisualStudio.Component.WebDeploy	Web Deploy	15.8.27729.1	Recommended
Microsoft.Net.Component.4.6.2.SDK	.NET Framework 4.6.2 SDK	15.6.27406.0	Optional
Microsoft.Net.Component.4.6.2.TargetingPack	.NET Framework 4.6.2 targeting pack	15.6.27406.0	Optional

Component ID	Name	Version	Dependency Type
Microsoft.Net.Component.4.7.1.SDK	.NET Framework 4.7.1 SDK	15.6.27406.0	Optional
Microsoft.Net.Component.4.7.1.TargetingPack	.NET Framework 4.7.1 targeting pack	15.6.27406.0	Optional
Microsoft.Net.Component.4.7.2.SDK	.NET Framework 4.7.2 SDK	15.8.27825.0	Optional
Microsoft.Net.Component.4.7.2.TargetingPack	.NET Framework 4.7.2 targeting pack	15.8.27825.0	Optional
Microsoft.Net.Component.4.7.SDK	.NET Framework 4.7 SDK	15.6.27406.0	Optional
Microsoft.Net.Component.4.7.TargetingPack	.NET Framework 4.7 targeting pack	15.6.27406.0	Optional
Microsoft.Net.ComponentGroup.4.6.2.DeveloperTools	.NET Framework 4.6.2 development tools	15.6.27406.0	Optional
Microsoft.Net.ComponentGroup.4.7.1.DeveloperTools	.NET Framework 4.7.1 development tools	15.6.27406.0	Optional
Microsoft.Net.ComponentGroup.4.7.2.DeveloperTools	.NET Framework 4.7.2 development tools	15.8.27825.0	Optional
Microsoft.Net.ComponentGroup.4.7.DeveloperTools	.NET Framework 4.7 development tools	15.6.27406.0	Optional

Python development

ID: Microsoft.VisualStudio.Workload.Python

Description: Editing, debugging, interactive development and source control for Python.

Components included by this workload

Component ID	Name	Version	Dependency Type
Microsoft.Component.PythonTools	Python language support	15.0.26823.1	Required
Component.CPython3.x64	Python 3 64-bit (3.6.6)	3.6.6	Recommended
Microsoft.Component.CookiecutterTools	Cookiecutter template support	15.0.26621.2	Recommended
Microsoft.Component.PythonTools.Web	Python web support	15.9.28107.0	Recommended
Microsoft.VisualStudio.Component.Common.Azure.Tools	Connectivity and publishing tools	15.9.28107.0	Recommended

Component ID	Name	Version	Dependency Type
Microsoft.VisualStudio.Component.JavaScript.TypeScript	JavaScript and TypeScript language support	15.9.28125.51	Recommended
Microsoft.VisualStudio.Component.SQL.CLR	CLR data types for SQL Server	15.0.26208.0	Recommended
Microsoft.VisualStudio.Component.TypeScript.3.1	TypeScript 3.1 SDK	15.0.28218.60	Recommended
Microsoft.VisualStudio.Component.VisualStudioData	Data sources and service references	15.6.27406.0	Recommended
Microsoft.VisualStudio.Component.WebDeploy	Web Deploy	15.8.27729.1	Recommended
Microsoft.VisualStudio.ComponentGroup.WebToolsExtensions	ASP.NET and web development	15.8.27825.0	Recommended
Component.Anaconda2.x64	Anaconda2 64-bit (5.2.0)	5.2.0	Optional
Component.Anaconda2.x86	Anaconda2 32-bit (5.2.0)	5.2.0	Optional
Component.Anaconda3.x64	Anaconda3 64-bit (5.2.0)	5.2.0	Optional
Component.Anaconda3.x86	Anaconda3 32-bit (5.2.0)	5.2.0	Optional
Component.CPython2.x64	Python 2 64-bit (2.7.14)	2.7.14	Optional
Component.CPython2.x86	Python 2 32-bit (2.7.14)	2.7.14	Optional
Component.CPython3.x86	Python 3 32-bit (3.6.6)	3.6.6	Optional
Component.Microsoft.VisualStudio.RazorExtension	Razor Language Services	15.0.26720.2	Optional
Component.Microsoft.Web.LibraryManager	Library Manager	15.8.27705.0	Optional
Component.WebSocket	WebSocket4Net	15.0.26606.0	Optional
Microsoft.Component.ClickOnce	ClickOnce Publishing	15.8.27825.0	Optional
Microsoft.Component_MSBUILD	MSBuild	15.7.27520.0	Optional
Microsoft.Component.NetFX.Native	.NET Native	15.0.26208.0	Optional
Microsoft.Component.PythonTools.UWP	Python IoT support	15.0.26606.0	Optional

Component ID	Name	Version	Dependency Type
Microsoft.Component.VC.Runtime.UCRTSDK	Windows Universal CRT SDK	15.6.27309.0	Optional
Microsoft.ComponentGroup.PythonTools.NativeDevelopment	Python native development tools	15.9.28307.102	Optional
Microsoft.Net.Component.4.5.2.TargetingPack	.NET Framework 4.5.2 targeting pack	15.6.27406.0	Optional
Microsoft.Net.Component.4.5.TargetingPack	.NET Framework 4.5 targeting pack	15.6.27406.0	Optional
Microsoft.Net.Component.4.6.1.SDK	.NET Framework 4.6.1 SDK	15.6.27406.0	Optional
Microsoft.Net.Component.4.6.1.TargetingPack	.NET Framework 4.6.1 targeting pack	15.6.27406.0	Optional
Microsoft.Net.ComponentGroup.DevelopmentPrerequisites	.NET Framework 4.6.1 development tools	15.8.27825.0	Optional
Microsoft.Net.Core.Component.SDK.2.1	.NET Core 2.1 development tools	15.8.27924.0	Optional
Microsoft.VisualStudio.Component.AppInsights.Tools	Developer Analytics tools	15.8.27825.0	Optional
Microsoft.VisualStudio.Component.Azure.AuthoringTools	Azure Authoring Tools	15.9.28307.421	Optional
Microsoft.VisualStudio.Component.Azure.ClientLibs	Azure libraries for .NET	15.0.26208.0	Optional
Microsoft.VisualStudio.Component.Azure.Compute.Emulator	Azure Compute Emulator	15.9.28307.421	Optional
Microsoft.VisualStudio.Component.Azure.Storage.Emulator	Azure Storage Emulator	15.9.28125.51	Optional
Microsoft.VisualStudio.Component.Azure.Waverton	Azure Cloud Services core tools	15.9.28107.0	Optional
Microsoft.VisualStudio.Component.Azure.Waverton.BuildTools	Azure Cloud Services build tools	15.7.27617.1	Optional
Microsoft.VisualStudio.Component.ClassDesigner	Class Designer	15.0.26208.0	Optional

Component ID	Name	Version	Dependency Type
Microsoft.VisualStudio.Component.DiagnosticTools	.NET profiling tools	15.8.27729.1	Optional
Microsoft.VisualStudio.Component.DockerTools	Container development tools	15.8.27906.1	Optional
Microsoft.VisualStudio.Component.DockerTools.BuildTools	Container development tools - Build Tools	15.7.27617.1	Optional
Microsoft.VisualStudio.Component.Graphics	Image and 3D model editors	15.6.27406.0	Optional
Microsoft.VisualStudio.Component.Graphics.Tools	Graphics debugger and GPU profiler for DirectX	15.6.27406.0	Optional
Microsoft.VisualStudio.Component.Graphics.Win81	Graphics Tools Windows 8.1 SDK	15.6.27406.0	Optional
Microsoft.VisualStudio.Component.IISExpress	IIS Express	15.0.26208.0	Optional
Microsoft.VisualStudio.Component.JavaScript.Diagnostics	JavaScript diagnostics	15.8.27729.1	Optional
Microsoft.VisualStudio.Component.ManagedDesktop.Core	Managed Desktop Workload Core	15.8.27729.1	Optional
Microsoft.VisualStudio.Component.NuGet	NuGet package manager	15.9.28016.0	Optional
Microsoft.VisualStudio.Component.PortableLibrary	.NET Portable Library targeting pack	15.6.27309.0	Optional
Microsoft.VisualStudio.Component.Roslyn.Compiler	C# and Visual Basic Roslyn compilers	15.6.27309.0	Optional
Microsoft.VisualStudio.Component.Roslyn.LanguageServices	C# and Visual Basic	15.8.27729.1	Optional
Microsoft.VisualStudio.Component.SQL.ADAL	SQL ADAL runtime	15.6.27406.0	Optional
Microsoft.VisualStudio.Component.SQL.CMDUtils	SQL Server Command Line Utilities	15.0.26208.0	Optional
Microsoft.VisualStudio.Component.SQL.DataSources	Data sources for SQL Server support	15.0.26621.2	Optional
Microsoft.VisualStudio.Component.SQL.LocalDB.Runtime	SQL Server Express 2016 LocalDB	15.7.27617.1	Optional

Component ID	Name	Version	Dependency Type
Microsoft.VisualStudio.Component.SQL.NCLI	SQL Server Native Client	15.0.26208.0	Optional
Microsoft.VisualStudio.Component.SQL.SSDT	SQL Server Data Tools	15.9.28107.0	Optional
Microsoft.VisualStudio.Component.Static.Analysis.Tools	Static analysis tools	15.0.26208.0	Optional
Microsoft.VisualStudio.Component.TextTemplating	Text Template Transformation	15.0.26208.0	Optional
Microsoft.VisualStudio.Component.VC.140	VC++ 2015.3 v14.00 (v140) toolset for desktop	15.7.27617.1	Optional
Microsoft.VisualStudio.Component.VC.CoreLDE	Visual Studio C++ core features	15.6.27406.0	Optional
Microsoft.VisualStudio.Component.VC.DiagnosticTools	C++ profiling tools	15.0.26823.1	Optional
Microsoft.VisualStudio.Component.VC.Tools.x86.x64	VC++ 2017 version 15.9 v14.16 latest v141 tools	15.9.28230.55	Optional
Microsoft.VisualStudio.Component.Web	ASP.NET and web development tools	15.8.27825.0	Optional
Microsoft.VisualStudio.Component.Windows10SDK	Windows Universal C Runtime	15.6.27406.0	Optional
Microsoft.VisualStudio.Component.Windows10SDK.1058.6	Windows 10 SDK (10.0.10586.0)	15.6.27406.0	Optional
Microsoft.VisualStudio.Component.Windows10SDK.1776.3	Windows 10 SDK (10.0.17763.0)	15.9.28307.102	Optional
Microsoft.VisualStudio.Component.Windows81SDK	Windows 8.1 SDK	15.6.27406.0	Optional
Microsoft.VisualStudio.ComponentGroup.Web	ASP.NET and web development tools prerequisites	15.9.28219.51	Optional

Universal Windows Platform development

ID: Microsoft.VisualStudio.Workload.Universal

Description: Create applications for the Universal Windows Platform with C#, VB, JavaScript, or optionally C++.

Components included by this workload

Component ID	Name	Version	Dependency Type
Component.WebSocket	WebSocket4Net	15.0.26606.0	Required
Microsoft.Component.ClickOnce	ClickOnce Publishing	15.8.27825.0	Required
Microsoft.Component.NetFX.Native	.NET Native	15.0.26208.0	Required
Microsoft.ComponentGroup.Blend	Blend for Visual Studio	15.6.27406.0	Required
Microsoft.Net.Component.4.5.TargetingPack	.NET Framework 4.5 targeting pack	15.6.27406.0	Required
Microsoft.Net.Component.4.6.1.SDK	.NET Framework 4.6.1 SDK	15.6.27406.0	Required
Microsoft.Net.Core.Component.SDK.2.1	.NET Core 2.1 development tools	15.8.27924.0	Required
Microsoft.VisualStudio.Component.AppInsights.Tools	Developer Analytics tools	15.8.27825.0	Required
Microsoft.VisualStudio.Component.DiagnosticTools	.NET profiling tools	15.8.27729.1	Required
Microsoft.VisualStudio.Component.Graphics	Image and 3D model editors	15.6.27406.0	Required
Microsoft.VisualStudio.Component.JavaScript.Diagnostics	JavaScript diagnostics	15.8.27729.1	Required
Microsoft.VisualStudio.Component.JavaScript.TypeScript	JavaScript and TypeScript language support	15.9.28125.51	Required
Microsoft.VisualStudio.Component.NuGet	NuGet package manager	15.9.28016.0	Required
Microsoft.VisualStudio.Component.PortableLibrary	.NET Portable Library targeting pack	15.6.27309.0	Required
Microsoft.VisualStudio.Component.Roslyn.Compiler	C# and Visual Basic Roslyn compilers	15.6.27309.0	Required
Microsoft.VisualStudio.Component.Roslyn.LanguageServices	C# and Visual Basic	15.8.27729.1	Required
Microsoft.VisualStudio.Component.SQL.CLR	CLR data types for SQL Server	15.0.26208.0	Required
Microsoft.VisualStudio.Component.Static.Analysis.Tools	Static analysis tools	15.0.26208.0	Required

Component ID	Name	Version	Dependency Type
Microsoft.VisualStudio.Component.TypeScript.3.1	TypeScript 3.1 SDK	15.0.28218.60	Required
Microsoft.VisualStudio.Component.UWP.Support	Universal Windows Platform tools	15.9.28119.51	Required
Microsoft.VisualStudio.Component.VisualStudioData	Data sources and service references	15.6.27406.0	Required
Microsoft.VisualStudio.Component.Windows10SDK.17763	Windows 10 SDK (10.0.17763.0)	15.9.28307.102	Required
Microsoft.VisualStudio.ComponentGroup.UWP.Cordova	Universal Windows Platform tools for Cordova	15.9.28307.102	Required
Microsoft.VisualStudio.ComponentGroup.UWP.NetCoreAndStandard	.NET Native and .NET Standard	15.8.27906.1	Required
Microsoft.VisualStudio.ComponentGroup.UWP.Xamarin	Universal Windows Platform tools for Xamarin	15.9.28307.102	Required
Microsoft.VisualStudio.ComponentGroup.WebToolsExtensions	ASP.NET and web development	15.8.27825.0	Required
Microsoft.Component.VC.Runtime.OSSupport	Visual C++ runtime for UWP	15.6.27406.0	Optional
Microsoft.Net.Component.4.7.2.SDK	.NET Framework 4.7.2 SDK	15.8.27825.0	Optional
Microsoft.VisualStudio.Component.Graphics.Tools	Graphics debugger and GPU profiler for DirectX	15.6.27406.0	Optional
Microsoft.VisualStudio.Component.Graphics.Win81	Graphics Tools Windows 8.1 SDK	15.6.27406.0	Optional
Microsoft.VisualStudio.Component.Phone Emulator.15254	Windows 10 Mobile Emulator (Fall Creators Update)	15.0.27406.0	Optional
Microsoft.VisualStudio.Component.UWP.VC.ARM64	C++ Universal Windows Platform tools for ARM64	15.0.28125.51	Optional
Microsoft.VisualStudio.Component.VC.Corelde	Visual Studio C++ core features	15.6.27406.0	Optional
Microsoft.VisualStudio.Component.VC.Tools.ARM	Visual C++ compilers and libraries for ARM	15.8.27825.0	Optional
Microsoft.VisualStudio.Component.VC.Tools.ARM64	Visual C++ compilers and libraries for ARM64	15.9.28230.55	Optional

Component ID	Name	Version	Dependency Type
Microsoft.VisualStudio.Component.VC.Tools.x86.x64	VC++ 2017 version 15.9 v14.16 latest v141 tools	15.9.28230.55	Optional
Microsoft.VisualStudio.Component.Windows10SDK.10240	Windows 10 SDK (10.0.10240.0)	15.6.27406.0	Optional
Microsoft.VisualStudio.Component.Windows10SDK.10586	Windows 10 SDK (10.0.10586.0)	15.6.27406.0	Optional
Microsoft.VisualStudio.Component.Windows10SDK.14393	Windows 10 SDK (10.0.14393.0)	15.6.27406.0	Optional
Microsoft.VisualStudio.Component.Windows10SDK.15063/Desktop	Windows 10 SDK (10.0.15063.0) for Desktop C++ [x86 and x64]	15.6.27406.0	Optional
Microsoft.VisualStudio.Component.Windows10SDK.15063/UWP	Windows 10 SDK (10.0.15063.0) for UWP: C#, VB, JS	15.6.27406.0	Optional
Microsoft.VisualStudio.Component.Windows10SDK.15063/UWP.Native	Windows 10 SDK (10.0.15063.0) for UWP: C++	15.6.27406.0	Optional
Microsoft.VisualStudio.Component.Windows10SDK.16299/Desktop	Windows 10 SDK (10.0.16299.0) for Desktop C++ [x86 and x64]	15.6.27406.0	Optional
Microsoft.VisualStudio.Component.Windows10SDK.16299/Desktop.arm	Windows 10 SDK (10.0.16299.0) for Desktop C++ [ARM and ARM64]	15.6.27406.0	Optional
Microsoft.VisualStudio.Component.Windows10SDK.16299/UWP	Windows 10 SDK (10.0.16299.0) for UWP: C#, VB, JS	15.6.27406.0	Optional
Microsoft.VisualStudio.Component.Windows10SDK.16299/UWP.Native	Windows 10 SDK (10.0.16299.0) for UWP: C++	15.6.27406.0	Optional
Microsoft.VisualStudio.Component.Windows10SDK.17134	Windows 10 SDK (10.0.17134.0)	15.9.28307.102	Optional
Microsoft.VisualStudio.Component.Windows10SDK.IpOverUsb	USB Device Connectivity	15.9.28307.102	Optional
Microsoft.VisualStudio.ComponentGroup.UWP.VC	C++ Universal Windows Platform tools	15.9.28307.102	Optional

Component ID	Name	Version	Dependency Type
Microsoft.VisualStudio.ComponentGroup.Windows10SDK.15063	Windows 10 SDK (10.0.15063.0)	15.8.27825.0	Optional
Microsoft.VisualStudio.ComponentGroup.Windows10SDK.16299	Windows 10 SDK (10.0.16299.0)	15.8.27825.0	Optional

Visual Studio extension development

ID: Microsoft.VisualStudio.Workload.VisualStudioExtension

Description: Create add-ons and extensions for Visual Studio, including new commands, code analyzers and tool windows.

Components included by this workload

Component ID	Name	Version	Dependency Type
Microsoft.Component.ClickOnce	ClickOnce Publishing	15.8.27825.0	Required
Microsoft.Component.MSBuild	MSBuild	15.7.27520.0	Required
Microsoft.Net.Component.4.6.1.SDK	.NET Framework 4.6.1 SDK	15.6.27406.0	Required
Microsoft.Net.Component.4.6.1.TargetingPack	.NET Framework 4.6.1 targeting pack	15.6.27406.0	Required
Microsoft.Net.Component.4.6.TargetingPack	.NET Framework 4.6 targeting pack	15.6.27406.0	Required
Microsoft.Net.ComponentGroup.DevelopmentPrerequisites	.NET Framework 4.6.1 development tools	15.8.27825.0	Required
Microsoft.VisualStudio.Component.NuGet	NuGet package manager	15.9.28016.0	Required
Microsoft.VisualStudio.Component.PortableLibrary	.NET Portable Library targeting pack	15.6.27309.0	Required
Microsoft.VisualStudio.Component.Roslyn.Compiler	C# and Visual Basic Roslyn compilers	15.6.27309.0	Required
Microsoft.VisualStudio.Component.Roslyn.LanguageServices	C# and Visual Basic	15.8.27729.1	Required
Microsoft.VisualStudio.Component.Static.Analysis.Tools	Static analysis tools	15.0.26208.0	Required

Component ID	Name	Version	Dependency Type
Microsoft.VisualStudio.Component.VSSDK	Visual Studio SDK	15.8.27729.1	Required
Microsoft.VisualStudio.ComponentGroup.VisualStudioExtension.Prerequisites	Visual Studio extension development prerequisites	15.7.27625.0	Required
Microsoft.VisualStudio.Component.DiagnosticTools	.NET profiling tools	15.8.27729.1	Recommended
Microsoft.VisualStudio.Component.TextTemplating	Text Template Transformation	15.0.26208.0	Recommended
Component.Dotfuscator	PreEmptive Protection - Dotfuscator	15.0.26208.0	Optional
Microsoft.Component.CodeAnalysis.SDK	.NET Compiler Platform SDK	15.0.27729.1	Optional
Microsoft.Component.VC.Runtime.OSSupport	Visual C++ runtime for UWP	15.6.27406.0	Optional
Microsoft.VisualStudio.Component.AppInsights.Tools	Developer Analytics tools	15.8.27825.0	Optional
Microsoft.VisualStudio.Component.ClassDesigner	Class Designer	15.0.26208.0	Optional
Microsoft.VisualStudio.Component.DslTools	Modeling SDK	15.0.27005.2	Optional
Microsoft.VisualStudio.Component.VC.ATL	Visual C++ ATL for x86 and x64	15.7.27625.0	Optional
Microsoft.VisualStudio.Component.VC.ATLMFC	Visual C++ MFC for x86 and x64	15.7.27625.0	Optional
Microsoft.VisualStudio.Component.VC.Corelde	Visual Studio C++ core features	15.6.27406.0	Optional
Microsoft.VisualStudio.Component.VC.Tools.x86.x64	VC++ 2017 version 15.9 v14.16 latest v141 tools	15.9.28230.55	Optional

Mobile development with JavaScript

ID: Microsoft.VisualStudio.Workload.WebCrossPlat

Description: Build Android, iOS and UWP apps using Tools for Apache Cordova.

Components included by this workload

Component ID	Name	Version	Dependency Type
Component.CordovaToolset.6.3.1	Cordova 6.3.1 toolset	15.7.27625.0	Required
Component.WebSocket	WebSocket4Net	15.0.26606.0	Required
Microsoft.VisualStudio.Component.Cordova	Mobile development with JavaScript core features	15.0.26606.0	Required
Microsoft.VisualStudio.Component.JavaScript.Diagnostics	JavaScript diagnostics	15.8.27729.1	Required
Microsoft.VisualStudio.Component.JavaScript.ProjectSystem	JavaScript ProjectSystem and Shared Tooling	15.0.26606.0	Required
Microsoft.VisualStudio.Component.JavaScript.TypeScript	JavaScript and TypeScript language support	15.9.28125.51	Required
Microsoft.VisualStudio.Component.TypeScript.2.3	TypeScript 2.3 SDK	15.8.27729.1	Required
Microsoft.VisualStudio.Component.TypeScript.3.1	TypeScript 3.1 SDK	15.0.28218.60	Required
Microsoft.VisualStudio.ComponentGroup.WebToolsExtensions	ASP.NET and web development	15.8.27825.0	Required
Component.Android.SDK23.Private	Android SDK setup (API level 23) (local install for Mobile development with JavaScript / C++)	15.9.28016.0	Optional
Component.Google.Android.Emulator.API23.Private	Google Android Emulator (API Level 23) (local install)	15.6.27413.0	Optional
Component.HAXM.Private	Intel Hardware Accelerated Execution Manager (HAXM) (local install)	15.9.28307.421	Optional
Component.OpenJDK	Microsoft distribution OpenJDK	15.9.28125.51	Optional
Microsoft.Component.ClickOnce	ClickOnce Publishing	15.8.27825.0	Optional
Microsoft.Component.NetFX.Native	.NET Native	15.0.26208.0	Optional
Microsoft.VisualStudio.Component.AppInsights.Tools	Developer Analytics tools	15.8.27825.0	Optional
Microsoft.VisualStudio.Component.DiagnosticTools	.NET profiling tools	15.8.27729.1	Optional

Component ID	Name	Version	Dependency Type
Microsoft.VisualStudio.Component.Git	Git for Windows	15.0.26208.0	Optional
Microsoft.VisualStudio.Component.Graphics	Image and 3D model editors	15.6.27406.0	Optional
Microsoft.VisualStudio.Component.Phone Emulator.15254	Windows 10 Mobile Emulator (Fall Creators Update)	15.0.27406.0	Optional
Microsoft.VisualStudio.Component.SQL.CLR	CLR data types for SQL Server	15.0.26208.0	Optional
Microsoft.VisualStudio.Component.VisualStudioData	Data sources and service references	15.6.27406.0	Optional
Microsoft.VisualStudio.Component.Windows10SDK.17763	Windows 10 SDK (10.0.17763.0)	15.9.28307.102	Optional
Microsoft.VisualStudio.ComponentGroup.UWP.Cordova	Universal Windows Platform tools for Cordova	15.9.28307.102	Optional

Unaffiliated components

These are components that are not included with any workload, but may be selected as an individual component.

Component ID	Name	Version
Component.Android.Emulator	Visual Studio Emulator for Android	15.6.27413.0
Component.Android.NDK.R11C	Android NDK (R11C)	11.3.14
Component.Android.NDK.R11C_3264	Android NDK (R11C) (32bit)	11.3.16
Component.Android.SDK23	Android SDK setup (API level 23) (global install)	15.9.28107.0
Component.Android.SDK25	Android SDK setup (API level 25)	15.9.28107.0
Component.GitHub.VisualStudio	GitHub extension for Visual Studio	2.5.2.2500
Component.Google.Android.Emulator.API23.V2	Google Android Emulator (API Level 23) (global install)	15.6.27413.0
Component.Google.Android.Emulator.API25	Google Android Emulator (API Level 25)	15.7.27604.0
Microsoft.Component.Blend.SDK.WPF	Blend for Visual Studio SDK for .NET	15.6.27406.0
Microsoft.Component.HelpViewer	Help Viewer	15.9.28307.421

Component ID	Name	Version
Microsoft.VisualStudio.Component.DependencyValidation.Community	Dependency Validation	15.0.26208.0
Microsoft.VisualStudio.Component.GraphDocument	DGML editor	15.0.27005.2
Microsoft.VisualStudio.Component.LinqToSql	LINQ to SQL tools	15.6.27406.0
Microsoft.VisualStudio.Component.Phone.Emulator	Windows 10 Mobile Emulator (Anniversary Edition)	15.6.27406.0
Microsoft.VisualStudio.Component.Phone.Emulator.15063	Windows 10 Mobile Emulator (Creators Update)	15.6.27406.0
Microsoft.VisualStudio.Component.Runtime.Node.x86.6.4.0	Runtime for components based on Node.js v6.4.0 (x86)	15.7.27617.1
Microsoft.VisualStudio.Component.Runtime.Node.x86.7.4.0	Runtime for components based on Node.js v7.4.0 (x86)	15.7.27617.1
Microsoft.VisualStudio.Component.TypeScript.2.0	TypeScript 2.0 SDK	15.8.27729.1
Microsoft.VisualStudio.Component.TypeScript.2.1	TypeScript 2.1 SDK	15.8.27729.1
Microsoft.VisualStudio.Component.TypeScript.2.2	TypeScript 2.2 SDK	15.8.27729.1
Microsoft.VisualStudio.Component.TypeScript.2.5	TypeScript 2.5 SDK	15.6.27406.0
Microsoft.VisualStudio.Component.TypeScript.2.6	TypeScript 2.6 SDK	15.0.27729.1
Microsoft.VisualStudio.Component.TypeScript.2.7	TypeScript 2.7 SDK	15.0.27729.1
Microsoft.VisualStudio.Component.TypeScript.2.8	TypeScript 2.8 SDK	15.0.27729.1
Microsoft.VisualStudio.Component.TypeScript.2.9	TypeScript 2.9 SDK	15.0.27924.0
Microsoft.VisualStudio.Component.TypeScript.3.0	TypeScript 3.0 SDK	15.0.27924.0
Microsoft.VisualStudio.Component.VC.ATL.ARM	Visual C++ ATL for ARM	15.7.27625.0
Microsoft.VisualStudio.Component.VC.ATL.ARM.Spectre	Visual C++ ATL for ARM with Spectre Mitigations	15.7.27625.0

Component ID	Name	Version
Microsoft.VisualStudio.Component.VC.ATL.ARM64	Visual C++ ATL for ARM64	15.7.27625.0
Microsoft.VisualStudio.Component.VC.ATL.ARM64.Spectre	Visual C++ ATL for ARM64 with Spectre Mitigations	15.7.27625.0
Microsoft.VisualStudio.Component.VC.ATL.Spectre	Visual C++ ATL (x86/x64) with Spectre Mitigations	15.7.27625.0
Microsoft.VisualStudio.Component.VC.ATLMFC.Spectre	Visual C++ MFC for x86/x64 with Spectre Mitigations	15.7.27625.0
Microsoft.VisualStudio.Component.VC.ClangC2	Clang/C2 (experimental)	15.7.27520.0
Microsoft.VisualStudio.Component.VC.MFC.ARM	Visual C++ MFC for ARM	15.7.27625.0
Microsoft.VisualStudio.Component.VC.MFC.ARM.Spectre	Visual C++ MFC for ARM with Spectre Mitigations	15.7.27625.0
Microsoft.VisualStudio.Component.VC.MFC.ARM64	Visual C++ MFC for ARM64	15.7.27625.0
Microsoft.VisualStudio.Component.VC.MFC.ARM64.Spectre	Visual C++ MFC support for ARM64 with Spectre Mitigations	15.7.27625.0
Microsoft.VisualStudio.Component.VC.Runtimes.ARM.Spectre	VC++ 2017 version 15.9 v14.16 Libs for Spectre (ARM)	15.9.28230.55
Microsoft.VisualStudio.Component.VC.Runtimes.ARM64.Spectre	VC++ 2017 version 15.9 v14.16 Libs for Spectre (ARM64)	15.9.28230.55
Microsoft.VisualStudio.Component.VC.Runtimes.x86.x64.Spectre	VC++ 2017 version 15.9 v14.16 Libs for Spectre (x86 and x64)	15.9.28230.55
Microsoft.VisualStudio.Component.VC.Tools.14.11	VC++ 2017 version 15.4 v14.11 toolset	15.0.27924.0
Microsoft.VisualStudio.Component.VC.Tools.14.12	VC++ 2017 version 15.5 v14.12 toolset	15.0.27924.0
Microsoft.VisualStudio.Component.VC.Tools.14.13	VC++ 2017 version 15.6 v14.13 toolset	15.0.27924.0
Microsoft.VisualStudio.Component.VC.Tools.14.14	VC++ 2017 version 15.7 v14.14 toolset	15.0.27924.0
Microsoft.VisualStudio.Component.VC.Tools.14.15	VC++ 2017 version 15.8 v14.15 toolset	15.0.28230.55

When you set dependencies in your VSIX manifest, you must specify Component IDs only. Use the tables on this page to determine the minimum component dependencies. In some scenarios, this might mean that you specify only one component from a workload. In other scenarios, it might mean that you specify multiple components.

from a single workload or multiple components from multiple workloads. For more information, see the [How to: Migrate extensibility projects to Visual Studio](#) page.

For more information about how to use these IDs, see the [Use command-line parameters to install Visual Studio](#) page. And, for a list of workload and component IDs for other products, see the [Visual Studio workload and component IDs](#) page.

Visual Studio core editor (included with Visual Studio Professional 2019)

ID: Microsoft.VisualStudio.Workload.CoreEditor

Description: The Visual Studio core shell experience, including syntax-aware code editing, source code control and work item management.

Components included by this workload

COMPONENT ID	NAME	VERSION	DEPENDENCY TYPE
Microsoft.VisualStudio.Component.CoreEditor	Visual Studio core editor	16.1.28811.260	Required
Microsoft.VisualStudio.Component.StartPageExperiment.Cpp	Visual Studio Start Page for C++ Users	16.0.28315.86	Optional

Azure development

ID: Microsoft.VisualStudio.Workload.Azure

Description: Azure SDKs, tools, and projects for developing cloud apps and creating resources using .NET Core and .NET Framework. Also includes tools for containerizing your application, including Docker support.

Components included by this workload

COMPONENT ID	NAME	VERSION	DEPENDENCY TYPE
Component.Microsoft.VisualStudio.RazorExtension	Razor Language Services	16.0.28714.129	Required
Component.Microsoft.VisualStudio.Web.AzureFunctions	Azure WebJobs Tools	16.0.28714.129	Required
Component.Microsoft.Web.LibraryManager	Library Manager	16.0.28315.86	Required
Microsoft.Component_MSBUILD	MSBuild	16.4.29429.68	Required
Microsoft.Net.Component.4.5.2.TargetingPack	.NET Framework 4.5.2 targeting pack	16.0.28517.75	Required
Microsoft.Net.Component.4.5.TargetingPack	.NET Framework 4.5 targeting pack	16.0.28517.75	Required

Component ID	Name	Version	Dependency Type
Microsoft.Net.Component.4.6.1.TargetingPack	.NET Framework 4.6.1 targeting pack	16.0.28517.75	Required
Microsoft.Net.Component.4.7.2.TargetingPack	.NET Framework 4.7.2 targeting pack	16.0.28517.75	Required
Microsoft.Net.Component.4.8.SDK	.NET Framework 4.8 SDK	16.4.29313.120	Required
Microsoft.Net.ComponentGroup.DevelopmentPrerequisites	.NET Framework 4.7.2 development tools	16.3.29207.166	Required
Microsoft.NetCore.Component.DevelopmentTools	.NET Core development tools	16.4.29511.114	Required
Microsoft.NetCore.Component.SDK	.NET Core 3.1 SDK	16.4.29519.181	Required
Microsoft.NetCore.Component.Web	.NET Core development tools	16.4.29511.114	Required
Microsoft.VisualStudio.Component.Azure.AuthoringTools	Azure Authoring Tools	16.0.28625.61	Required
Microsoft.VisualStudio.Component.Azure.ClientLibs	Azure libraries for .NET	16.0.28315.86	Required
Microsoft.VisualStudio.Component.Azure.Compute.Emulator	Azure Compute Emulator	16.1.28810.153	Required
Microsoft.VisualStudio.Component.Azure.Storage.Emulator	Azure Storage Emulator	16.4.29313.120	Required
Microsoft.VisualStudio.Component.CloudExplorer	Cloud Explorer	16.0.28625.61	Required
Microsoft.VisualStudio.Component.Common.Azure.Tools	Connectivity and publishing tools	16.4.29409.204	Required
Microsoft.VisualStudio.Component.DockerTools	Container development tools	16.4.29409.204	Required
Microsoft.VisualStudio.Component.FSharp	F# language support	16.0.28315.86	Required
Microsoft.VisualStudio.Component.FSharp.WebTemplates	F# language support for web projects	16.3.29207.166	Required
Microsoft.VisualStudio.Component.IIExpress	IIS Express	16.0.28315.86	Required

Component ID	Name	Version	Dependency Type
Microsoft.VisualStudio.Component.IntelliCode	IntelliCode	0.1	Required
Microsoft.VisualStudio.Component.JavaScript.Diagnostics	JavaScript diagnostics	16.0.28517.75	Required
Microsoft.VisualStudio.Component.JavaScript.TypeScript	JavaScript and TypeScript language support	16.4.29409.204	Required
Microsoft.VisualStudio.Component.ManagedDesktop.Core	Managed Desktop Workload Core	16.4.29318.151	Required
Microsoft.VisualStudio.Component.MSODBC.SQL	SQL Server ODBC Driver	16.0.28625.61	Required
Microsoft.VisualStudio.Component.MSSQL.CMDLnUtils	SQL Server Command Line Utilities	16.0.28707.177	Required
Microsoft.VisualStudio.Component.NuGet	NuGet package manager	16.1.28829.92	Required
Microsoft.VisualStudio.Component.Roslyn.Compiler	C# and Visual Basic Roslyn compilers	16.0.28714.129	Required
Microsoft.VisualStudio.Component.Roslyn.LanguageServices	C# and Visual Basic	16.4.29429.68	Required
Microsoft.VisualStudio.Component.SQL.ADAL	SQL ADAL runtime	16.0.28517.75	Required
Microsoft.VisualStudio.Component.SQL.CLR	CLR data types for SQL Server	16.0.28315.86	Required
Microsoft.VisualStudio.Component.SQL.DataSources	Data sources for SQL Server support	16.0.28315.86	Required
Microsoft.VisualStudio.Component.SQL.LocalDB.Runtime	SQL Server Express 2016 LocalDB	16.0.28625.61	Required
Microsoft.VisualStudio.Component.SQL.SSDT	SQL Server Data Tools	16.3.29207.166	Required
Microsoft.VisualStudio.Component.TextTemplating	Text Template Transformation	16.0.28625.61	Required
Microsoft.VisualStudio.Component.TypeScript.3.7	TypeScript 3.7 SDK	16.0.29429.68	Required
Microsoft.VisualStudio.Component.Web	ASP.NET and web development tools	16.0.28517.75	Required

Component ID	Name	Version	Dependency Type
Microsoft.VisualStudio.ComponentGroup.Azure.Prerequisites	Azure development prerequisites	16.4.29409.204	Required
Microsoft.VisualStudio.ComponentGroup.AzureFunctions	Azure WebJobs Tools	16.0.28621.142	Required
Microsoft.VisualStudio.ComponentGroup.Web	ASP.NET and web development tools prerequisites	16.4.29318.151	Required
Microsoft.VisualStudio.ComponentGroup.WebToolsExtensions	ASP.NET and web development	16.0.28621.142	Required
Microsoft.Component.Azure.DataLake.Tools	Azure Data Lake and Stream Analytics Tools	16.4.29313.120	Recommended
Microsoft.Net.Component.4.5.1.TargetingPack	.NET Framework 4.5.1 targeting pack	16.0.28517.75	Recommended
Microsoft.Net.Component.4.6.TargetingPack	.NET Framework 4.6 targeting pack	16.0.28517.75	Recommended
Microsoft.Net.Component.4.TargetingPack	.NET Framework 4 targeting pack	16.0.28517.75	Recommended
Microsoft.Net.ComponentGroup.TargetingPacks.Common	.NET Framework 4 – 4.6 development tools	16.0.28516.191	Recommended
Microsoft.Net.Core.Component.SDK.2.1	.NET Core 2.1 LTS Runtime	16.4.29519.181	Recommended
Microsoft.VisualStudio.Component.AspNet45	Advanced ASP.NET features	16.0.28315.86	Recommended
Microsoft.VisualStudio.Component.Azure.Kubernetes.Tools	Visual Studio Tools for Kubernetes	16.0.28625.61	Recommended
Microsoft.VisualStudio.Component.Azure.PowerShell	Azure Powershell	16.4.29429.68	Recommended
Microsoft.VisualStudio.Component.Azure.ResourceManager.Tools	Azure Resource Manager core tools	16.4.29409.204	Recommended
Microsoft.VisualStudio.Component.Azure.ServiceFabric.Tools	Service Fabric Tools	16.4.29313.120	Recommended
Microsoft.VisualStudio.Component.Azure.Waverton	Azure Cloud Services core tools	16.4.29409.204	Recommended

Component ID	Name	Version	Dependency Type
Microsoft.VisualStudio.Component.Azure.Waverton.BuildTools	Azure Cloud Services build tools	16.3.29207.166	Recommended
Microsoft.VisualStudio.Component.DiagnosticTools	.NET profiling tools	16.4.29429.68	Recommended
Microsoft.VisualStudio.Component.WebDeploy	Web Deploy	16.0.28517.75	Recommended
Microsoft.VisualStudio.ComponentGroup.Azure.CloudServices	Azure Cloud Services tools	16.4.29409.204	Recommended
Microsoft.VisualStudio.ComponentGroup.Azure.ResourceManager.Tools	Azure Resource Manager tools	16.0.28528.71	Recommended
Microsoft.Net.Component.4.6.2.TargetingPack	.NET Framework 4.6.2 targeting pack	16.0.28517.75	Optional
Microsoft.Net.Component.4.7.1.TargetingPack	.NET Framework 4.7.1 targeting pack	16.0.28517.75	Optional
Microsoft.Net.Component.4.7.TargetingPack	.NET Framework 4.7 targeting pack	16.0.28517.75	Optional
Microsoft.Net.Component.4.8.TargetingPack	.NET Framework 4.8 targeting pack	16.4.29313.120	Optional
Microsoft.Net.ComponentGroup.4.6.1.DeveloperTools	.NET Framework 4.6.1 development tools	16.3.29207.166	Optional
Microsoft.Net.ComponentGroup.4.6.2.DeveloperTools	.NET Framework 4.6.2 development tools	16.3.29207.166	Optional
Microsoft.Net.ComponentGroup.4.7.1.DeveloperTools	.NET Framework 4.7.1 development tools	16.3.29207.166	Optional
Microsoft.Net.ComponentGroup.4.7.DeveloperTools	.NET Framework 4.7 development tools	16.3.29207.166	Optional
Microsoft.Net.ComponentGroup.4.8.DeveloperTools	.NET Framework 4.8 development tools	16.4.29318.151	Optional
Microsoft.VisualStudio.Component.Azure.Storage.AzCopy	Azure Storage AzCopy	16.0.28517.75	Optional
Microsoft.VisualStudio.Component.Wcf.Tooling	Windows Communication Foundation	16.0.28625.61	Optional

Data storage and processing

ID: Microsoft.VisualStudio.Workload.Data

Description: Connect, develop, and test data solutions with SQL Server, Azure Data Lake, or Hadoop.

Components included by this workload

Component ID	Name	Version	Dependency Type
Component.Microsoft.VisualStudio.RazorExtension	Razor Language Services	16.0.28714.129	Recommended
Component.Microsoft.Web.LibraryManager	Library Manager	16.0.28315.86	Recommended
Microsoft.Component.Azure.DataLake.Tools	Azure Data Lake and Stream Analytics Tools	16.4.29313.120	Recommended
Microsoft.Component.MSBuild	MSBuild	16.4.29429.68	Recommended
Microsoft.Net.Component.4.5.1.TargetingPack	.NET Framework 4.5.1 targeting pack	16.0.28517.75	Recommended
Microsoft.Net.Component.4.5.2.TargetingPack	.NET Framework 4.5.2 targeting pack	16.0.28517.75	Recommended
Microsoft.Net.Component.4.5.TargetingPack	.NET Framework 4.5 targeting pack	16.0.28517.75	Recommended
Microsoft.Net.Component.4.6.TargetingPack	.NET Framework 4.6 targeting pack	16.0.28517.75	Recommended
Microsoft.Net.Component.4.7.2.TargetingPack	.NET Framework 4.7.2 targeting pack	16.0.28517.75	Recommended
Microsoft.Net.Component.4.8.SDK	.NET Framework 4.8 SDK	16.4.29313.120	Recommended
Microsoft.Net.Component.4.TargetingPack	.NET Framework 4 targeting pack	16.0.28517.75	Recommended
Microsoft.Net.ComponentGroup.DevelopmentPrerequisites	.NET Framework 4.7.2 development tools	16.3.29207.166	Recommended
Microsoft.Net.ComponentGroup.TargetingPacks.Common	.NET Framework 4 – 4.6 development tools	16.0.28516.191	Recommended
Microsoft.NetCore.Component.SDK	.NET Core 3.1 SDK	16.4.29519.181	Recommended
Microsoft.VisualStudio.Component.Azure.AuthoringTools	Azure Authoring Tools	16.0.28625.61	Recommended
Microsoft.VisualStudio.Component.Azure.ClientLibs	Azure libraries for .NET	16.0.28315.86	Recommended

Component ID	Name	Version	Dependency Type
Microsoft.VisualStudio.Component.Azure.Compute Emulator	Azure Compute Emulator	16.1.28810.153	Recommended
Microsoft.VisualStudio.Component.Azure.Storage.Emulator	Azure Storage Emulator	16.4.29313.120	Recommended
Microsoft.VisualStudio.Component.Azure.Waverton	Azure Cloud Services core tools	16.4.29409.204	Recommended
Microsoft.VisualStudio.Component.Azure.Waverton.Build Tools	Azure Cloud Services build tools	16.3.29207.166	Recommended
Microsoft.VisualStudio.Component.CloudExplorer	Cloud Explorer	16.0.28625.61	Recommended
Microsoft.VisualStudio.Component.Common.Azure.Tools	Connectivity and publishing tools	16.4.29409.204	Recommended
Microsoft.VisualStudio.Component.DockerTools	Container development tools	16.4.29409.204	Recommended
Microsoft.VisualStudio.Component.IISExpress	IIS Express	16.0.28315.86	Recommended
Microsoft.VisualStudio.Component.JavaScript.Diagnostics	JavaScript diagnostics	16.0.28517.75	Recommended
Microsoft.VisualStudio.Component.JavaScript.TypeScript	JavaScript and TypeScript language support	16.4.29409.204	Recommended
Microsoft.VisualStudio.Component.ManagedDesktop.Core	Managed Desktop Workload Core	16.4.29318.151	Recommended
Microsoft.VisualStudio.Component.MSODBC.SQL	SQL Server ODBC Driver	16.0.28625.61	Recommended
Microsoft.VisualStudio.Component.MSSQL.CMDLnUtils	SQL Server Command Line Utilities	16.0.28707.177	Recommended
Microsoft.VisualStudio.Component.NuGet	NuGet package manager	16.1.28829.92	Recommended
Microsoft.VisualStudio.Component.Roslyn.Compiler	C# and Visual Basic Roslyn compilers	16.0.28714.129	Recommended
Microsoft.VisualStudio.Component.Roslyn.LanguageServices	C# and Visual Basic	16.4.29429.68	Recommended

Component ID	Name	Version	Dependency Type
Microsoft.VisualStudio.Component.SQL.ADAL	SQL ADAL runtime	16.0.28517.75	Recommended
Microsoft.VisualStudio.Component.SQL.CLR	CLR data types for SQL Server	16.0.28315.86	Recommended
Microsoft.VisualStudio.Component.SQL.DataSources	Data sources for SQL Server support	16.0.28315.86	Recommended
Microsoft.VisualStudio.Component.SQL.LocalDB.Runtime	SQL Server Express 2016 LocalDB	16.0.28625.61	Recommended
Microsoft.VisualStudio.Component.SQL.SSDT	SQL Server Data Tools	16.3.29207.166	Recommended
Microsoft.VisualStudio.Component.TextTemplating	Text Template Transformation	16.0.28625.61	Recommended
Microsoft.VisualStudio.Component.TypeScript.3.7	TypeScript 3.7 SDK	16.0.29429.68	Recommended
Microsoft.VisualStudio.Component.Web	ASP.NET and web development tools	16.0.28517.75	Recommended
Microsoft.VisualStudio.ComponentGroup.Web	ASP.NET and web development tools prerequisites	16.4.29318.151	Recommended
Microsoft.VisualStudio.ComponentGroup.WebToolsExtensions	ASP.NET and web development	16.0.28621.142	Recommended
Microsoft.Net.Component.4.6.1.TargetingPack	.NET Framework 4.6.1 targeting pack	16.0.28517.75	Optional
Microsoft.VisualStudio.Component.FSharp.Desktop	F# desktop language support	16.0.28315.86	Optional

Data science and analytical applications

ID: Microsoft.VisualStudio.Workload.DataScience

Description: Languages and tooling for creating data science applications, including Python and F#.

Components included by this workload

Component ID	Name	Version	Dependency Type
Microsoft.Component.PythonTools	Python language support	16.4.29429.68	Recommended
Microsoft.Component.PythonTools.MinicondaX64	Python miniconda	16.2.29003.222	Recommended

Component ID	Name	Version	Dependency Type
Microsoft.Component.PythonTools.Web	Python web support	16.0.28517.75	Recommended
Microsoft.Net.Component.4.6.1.TargetingPack	.NET Framework 4.6.1 targeting pack	16.0.28517.75	Recommended
Microsoft.VisualStudio.Component.Common.Azure.Tools	Connectivity and publishing tools	16.4.29409.204	Recommended
Microsoft.VisualStudio.Component.FSharp.Desktop	F# desktop language support	16.0.28315.86	Recommended
Microsoft.VisualStudio.Component.JavaScript.TypeScript	JavaScript and TypeScript language support	16.4.29409.204	Recommended
Microsoft.VisualStudio.Component.NuGet	NuGet package manager	16.1.28829.92	Recommended
Microsoft.VisualStudio.Component.Roslyn.Compiler	C# and Visual Basic Roslyn compilers	16.0.28714.129	Recommended
Microsoft.VisualStudio.Component.Roslyn.LanguageServices	C# and Visual Basic	16.4.29429.68	Recommended
Microsoft.VisualStudio.Component.TypeScript.3.7	TypeScript 3.7 SDK	16.0.29429.68	Recommended
Microsoft.VisualStudio.Component.WebDeploy	Web Deploy	16.0.28517.75	Recommended
Microsoft.VisualStudio.ComponentGroup.WebToolsExtensions	ASP.NET and web development	16.0.28621.142	Recommended
Microsoft.ComponentGroup.PythonTools.NativeDevelopment	Python native development tools	16.2.29020.229	Optional
Microsoft.VisualStudio.Component.Graphics.Tools	Graphics debugger and GPU profiler for DirectX	16.0.28625.61	Optional
Microsoft.VisualStudio.Component.VC.Corelde	C++ core features	16.0.28625.61	Optional
Microsoft.VisualStudio.Component.VC.DiagnosticTools	C++ profiling tools	16.4.29429.68	Optional
Microsoft.VisualStudio.Component.VC.Tools.x86.x64	MSVC v142 - VS 2019 C++ x64/x86 build tools (v14.24)	16.4.29409.204	Optional
Microsoft.VisualStudio.Component.Windows10SDK	Windows Universal C Runtime	16.4.29409.204	Optional

Component ID	Name	Version	Dependency Type
Microsoft.VisualStudio.Component.Windows10SDK.18362	Windows 10 SDK (10.0.18362.0)	16.1.28829.92	Optional

.NET desktop development

ID: Microsoft.VisualStudio.Workload.ManagedDesktop

Description: Build WPF, Windows Forms, and console applications using C#, Visual Basic, and F# with .NET Core and .NET Framework.

Components included by this workload

Component ID	Name	Version	Dependency Type
Microsoft.Component.MSBuild	MSBuild	16.4.29429.68	Required
Microsoft.Net.Component.4.7.2.TargetingPack	.NET Framework 4.7.2 targeting pack	16.0.28517.75	Required
Microsoft.Net.Component.4.8.SDK	.NET Framework 4.8 SDK	16.4.29313.120	Required
Microsoft.Net.ComponentGroup.DevelopmentPrerequisites	.NET Framework 4.7.2 development tools	16.3.29207.166	Required
Microsoft.NetCore.Component.SDK	.NET Core 3.1 SDK	16.4.29519.181	Required
Microsoft.VisualStudio.Component.IntelliCode	IntelliCode	0.1	Required
Microsoft.VisualStudio.Component.ManagedDesktop.Core	Managed Desktop Workload Core	16.4.29318.151	Required
Microsoft.VisualStudio.Component.ManagedDesktop.Prerequisites	.NET desktop development tools	16.4.29429.68	Required
Microsoft.VisualStudio.Component.Roslyn.Compiler	C# and Visual Basic Roslyn compilers	16.0.28714.129	Required
Microsoft.VisualStudio.Component.Roslyn.LanguageServices	C# and Visual Basic	16.4.29429.68	Required
Microsoft.VisualStudio.Component.SQL CLR	CLR data types for SQL Server	16.0.28315.86	Required
Microsoft.VisualStudio.Component.TextTemplating	Text Template Transformation	16.0.28625.61	Required

Component ID	Name	Version	Dependency Type
Component.Microsoft.VisualStudio.LiveShare	Live Share	1.0.1076	Recommended
Microsoft.ComponentGroup.Blend	Blend for Visual Studio	16.0.28315.86	Recommended
Microsoft.Net.Component.4.5.1.TargetingPack	.NET Framework 4.5.1 targeting pack	16.0.28517.75	Recommended
Microsoft.Net.Component.4.5.2.TargetingPack	.NET Framework 4.5.2 targeting pack	16.0.28517.75	Recommended
Microsoft.Net.Component.4.5.TargetingPack	.NET Framework 4.5 targeting pack	16.0.28517.75	Recommended
Microsoft.Net.Component.4.6.1.TargetingPack	.NET Framework 4.6.1 targeting pack	16.0.28517.75	Recommended
Microsoft.Net.Component.4.6.TargetingPack	.NET Framework 4.6 targeting pack	16.0.28517.75	Recommended
Microsoft.Net.Component.4.TargetingPack	.NET Framework 4 targeting pack	16.0.28517.75	Recommended
Microsoft.Net.ComponentGroup.TargetingPacks.Common	.NET Framework 4 – 4.6 development tools	16.0.28516.191	Recommended
Microsoft.Net.Core.Component.SDK.2.1	.NET Core 2.1 LTS Runtime	16.4.29519.181	Recommended
Microsoft.NetCore.Component.DevelopmentTools	.NET Core development tools	16.4.29511.114	Recommended
Microsoft.VisualStudio.Component.Debugger.JustInTime	Just-In-Time debugger	16.0.28517.75	Recommended
Microsoft.VisualStudio.Component.DiagnosticTools	.NET profiling tools	16.4.29429.68	Recommended
Microsoft.VisualStudio.Component.EntityFramework	Entity Framework 6 tools	16.0.28315.86	Recommended
Microsoft.VisualStudio.Component.FSharp	F# language support	16.0.28315.86	Recommended
Microsoft.VisualStudio.Component.NuGet	NuGet package manager	16.1.28829.92	Recommended
Component.Dotfuscator	PreEmptive Protection - Dotfuscator	16.0.28528.71	Optional
Component.Microsoft.VisualStudio.RazorExtension	Razor Language Services	16.0.28714.129	Optional

Component ID	Name	Version	Dependency Type
Component.Microsoft.Web.LibraryManager	Library Manager	16.0.28315.86	Optional
Microsoft.Net.Component.4.6.2.TargetingPack	.NET Framework 4.6.2 targeting pack	16.0.28517.75	Optional
Microsoft.Net.Component.4.7.1.TargetingPack	.NET Framework 4.7.1 targeting pack	16.0.28517.75	Optional
Microsoft.Net.Component.4.7.TargetingPack	.NET Framework 4.7 targeting pack	16.0.28517.75	Optional
Microsoft.Net.Component.4.8.TargetingPack	.NET Framework 4.8 targeting pack	16.4.29313.120	Optional
Microsoft.Net.ComponentGroup.4.6.1.DeveloperTools	.NET Framework 4.6.1 development tools	16.3.29207.166	Optional
Microsoft.Net.ComponentGroup.4.6.2.DeveloperTools	.NET Framework 4.6.2 development tools	16.3.29207.166	Optional
Microsoft.Net.ComponentGroup.4.7.1.DeveloperTools	.NET Framework 4.7.1 development tools	16.3.29207.166	Optional
Microsoft.Net.ComponentGroup.4.7.DeveloperTools	.NET Framework 4.7 development tools	16.3.29207.166	Optional
Microsoft.Net.ComponentGroup.4.8.DeveloperTools	.NET Framework 4.8 development tools	16.4.29318.151	Optional
Microsoft.VisualStudio.Component.Common.Azure.Tools	Connectivity and publishing tools	16.4.29409.204	Optional
Microsoft.VisualStudio.Component.DockerTools	Container development tools	16.4.29409.204	Optional
Microsoft.VisualStudio.Component.FSharp.Desktop	F# desktop language support	16.0.28315.86	Optional
Microsoft.VisualStudio.Component.IISExpress	IIS Express	16.0.28315.86	Optional
Microsoft.VisualStudio.Component.JavaScript.Diagnostics	JavaScript diagnostics	16.0.28517.75	Optional
Microsoft.VisualStudio.Component.JavaScript.TypeScript	JavaScript and TypeScript language support	16.4.29409.204	Optional
Microsoft.VisualStudio.Component.MSODBC.SQL	SQL Server ODBC Driver	16.0.28625.61	Optional
Microsoft.VisualStudio.Component.MSSQL.CMDLnUtils	SQL Server Command Line Utilities	16.0.28707.177	Optional

Component ID	Name	Version	Dependency Type
Microsoft.VisualStudio.Component.PortableLibrary	.NET Portable Library targeting pack	16.0.28517.75	Optional
Microsoft.VisualStudio.Component.SQL.ADAL	SQL ADAL runtime	16.0.28517.75	Optional
Microsoft.VisualStudio.Component.SQL.DataSources	Data sources for SQL Server support	16.0.28315.86	Optional
Microsoft.VisualStudio.Component.SQL.LocalDB.Runtime	SQL Server Express 2016 LocalDB	16.0.28625.61	Optional
Microsoft.VisualStudio.Component.SQL.SSDT	SQL Server Data Tools	16.3.29207.166	Optional
Microsoft.VisualStudio.Component.TypeScript.3.7	TypeScript 3.7 SDK	16.0.29429.68	Optional
Microsoft.VisualStudio.Component.Wcf.Tooling	Windows Communication Foundation	16.0.28625.61	Optional
Microsoft.VisualStudio.Component.Web	ASP.NET and web development tools	16.0.28517.75	Optional
Microsoft.VisualStudio.Component.Windows10SDK.18362	Windows 10 SDK (10.0.18362.0)	16.1.28829.92	Optional
Microsoft.VisualStudio.ComponentGroup.MSIX.Packaging	MSIX Packaging Tools	16.4.29409.204	Optional
Microsoft.VisualStudio.ComponentGroup.Web	ASP.NET and web development tools prerequisites	16.4.29318.151	Optional
Microsoft.VisualStudio.ComponentGroup.WebToolsExtensions	ASP.NET and web development	16.0.28621.142	Optional

Game development with Unity

ID: Microsoft.VisualStudio.Workload.ManagedGame

Description: Create 2D and 3D games with Unity, a powerful cross-platform development environment.

Components included by this workload

Component ID	Name	Version	Dependency Type
Microsoft.Net.Component.3.5.DeveloperTools	.NET Framework 3.5 development tools	16.0.28517.75	Required
Microsoft.Net.Component.4.7.1.TargetingPack	.NET Framework 4.7.1 targeting pack	16.0.28517.75	Required

Component ID	Name	Version	Dependency Type
Microsoft.VisualStudio.Component.NuGet	NuGet package manager	16.1.28829.92	Required
Microsoft.VisualStudio.Component.Roslyn.Compiler	C# and Visual Basic Roslyn compilers	16.0.28714.129	Required
Microsoft.VisualStudio.Component.Roslyn.LanguageServices	C# and Visual Basic	16.4.29429.68	Required
Microsoft.VisualStudio.Component.Unity	Visual Studio Tools for Unity	16.0.28315.86	Required
Component.UnityEngine.x64	Unity 2019.2 64-bit Editor	16.4.29429.68	Recommended
Component.UnityEngine.x86	Unity 5.6 32-bit Editor	16.1.28811.260	Recommended

Linux development with C++

ID: Microsoft.VisualStudio.Workload.NativeCrossPlat

Description: Create and debug applications running in a Linux environment.

Components included by this workload

Component ID	Name	Version	Dependency Type
Component.MDD.Linux	C++ for Linux Development	16.4.29511.114	Required
Microsoft.VisualStudio.Component.IntelliCode	IntelliCode	0.1	Required
Microsoft.VisualStudio.Component.VC.Corelde	C++ core features	16.0.28625.61	Required
Component.Linux.CMake	C++ CMake tools for Linux	16.2.29003.222	Recommended
Microsoft.VisualStudio.ComponentGroup.WebToolsExtensions	ASP.NET and web development	16.0.28621.142	Recommended
Component.MDD.Linux.GCC.arm	Embedded and IoT development tools	16.4.29429.68	Optional

Desktop development with C++

ID: Microsoft.VisualStudio.Workload.NativeDesktop

Description: Build modern C++ apps for Windows using tools of your choice, including MSVC, Clang, CMake, or MSBuild.

Components included by this workload

Component ID	Name	Version	Dependency Type
Microsoft.Component.MSBuild	MSBuild	16.4.29429.68	Required
Microsoft.VisualStudio.Component.IntelliCode	IntelliCode	0.1	Required
Microsoft.VisualStudio.Component.Roslyn.Compiler	C# and Visual Basic Roslyn compilers	16.0.28714.129	Required
Microsoft.VisualStudio.Component.TextTemplating	Text Template Transformation	16.0.28625.61	Required
Microsoft.VisualStudio.Component.VC.CoreRedistributable	C++ core features	16.0.28625.61	Required
Microsoft.VisualStudio.Component.VC.Redist.14.Latest	C++ 2019 Redistributable Update	16.4.29429.68	Required
Microsoft.VisualStudio.ComponentGroup.NativeDesktop.Core	C++ core desktop features	16.2.29012.281	Required
Component.Microsoft.VisualStudio.LiveShare	Live Share	1.0.1076	Recommended
Microsoft.VisualStudio.Component.Debugger.JustInTime	Just-In-Time debugger	16.0.28517.75	Recommended
Microsoft.VisualStudio.Component.Graphics.Tools	Graphics debugger and GPU profiler for DirectX	16.0.28625.61	Recommended
Microsoft.VisualStudio.Component.NuGet	NuGet package manager	16.1.28829.92	Recommended
Microsoft.VisualStudio.Component.VC.ASAN	C++ AddressSanitizer (Experimental)	16.4.29429.68	Recommended
Microsoft.VisualStudio.Component.VC.ATL	C++ ATL for latest v142 build tools (x86 & x64)	16.4.29313.120	Recommended
Microsoft.VisualStudio.Component.VC.CMake.Project	C++ CMake tools for Windows	16.3.29103.31	Recommended
Microsoft.VisualStudio.Component.VC.DiagnosticTools	C++ profiling tools	16.4.29429.68	Recommended
Microsoft.VisualStudio.Component.VC.TestAdapterForBoostTest	Test Adapter for Boost.Test	16.0.28517.75	Recommended
Microsoft.VisualStudio.Component.VC.TestAdapterForGoogleTest	Test Adapter for Google Test	16.0.28517.75	Recommended

Component ID	Name	Version	Dependency Type
Microsoft.VisualStudio.Component.VC.Tools.x86.x64	MSVC v142 - VS 2019 C++ x64/x86 build tools (v14.24)	16.4.29409.204	Recommended
Microsoft.VisualStudio.Component.Windows10SDK.18362	Windows 10 SDK (10.0.18362.0)	16.1.28829.92	Recommended
Microsoft.VisualStudio.ComponentGroup.WebToolsExtensions.CMake	JSON editor	16.3.29207.166	Recommended
Component.Incredibuild	Incredibuild - Build Acceleration	16.0.28528.71	Optional
Component.IncredibuildMenu	IncredibuildMenu	1.5.0.10	Optional
Microsoft.Component.VC.Runtime.UCRTSDK	Windows Universal CRT SDK	16.0.28625.61	Optional
Microsoft.Net.Component.4.6.1.TargetingPack	.NET Framework 4.6.1 targeting pack	16.0.28517.75	Optional
Microsoft.Net.Component.4.8.SDK	.NET Framework 4.8 SDK	16.4.29313.120	Optional
Microsoft.VisualStudio.Component.JavaScript.Diagnostics	JavaScript diagnostics	16.0.28517.75	Optional
Microsoft.VisualStudio.Component.VC.140	MSVC v140 - VS 2015 C++ build tools (v14.00)	16.0.28625.61	Optional
Microsoft.VisualStudio.Component.VC.ATLMFC	C++ MFC for latest v142 build tools (x86 & x64)	16.4.29313.120	Optional
Microsoft.VisualStudio.Component.VC.CLI.Support	C++/CLI support for v142 build tools (14.24)	16.4.29409.204	Optional
Microsoft.VisualStudio.Component.VC.Llvm.Clang	C++ Clang Compiler for Windows (9.0.0)	16.4.29511.114	Optional
Microsoft.VisualStudio.Component.VC.Llvm.ClangToolset	C++ Clang-cl for v142 build tools (x64/x86)	16.3.29207.166	Optional
Microsoft.VisualStudio.Component.VC.Modules.x86.x64	C++ Modules for v142 build tools (x64/x86 – experimental)	16.0.28625.61	Optional
Microsoft.VisualStudio.Component.VC.v141.x86.x64	MSVC v141 - VS 2017 C++ x64/x86 build tools (v14.16)	16.1.28829.92	Optional
Microsoft.VisualStudio.Component.Windows10SDK.16299.9	Windows 10 SDK (10.0.16299.0)	16.0.28517.75	Optional

Component ID	Name	Version	Dependency Type
Microsoft.VisualStudio.Component.Windows10SDK.17134	Windows 10 SDK (10.0.17134.0)	16.0.28517.75	Optional
Microsoft.VisualStudio.Component.Windows10SDK.17763	Windows 10 SDK (10.0.17763.0)	16.0.28517.75	Optional
Microsoft.VisualStudio.ComponentGroup.NativeDesktop.Llvm.Clang	C++ Clang tools for Windows (9.0.0 - x64/x86)	16.4.29511.114	Optional

Game development with C++

ID: Microsoft.VisualStudio.Workload.NativeGame

Description: Use the full power of C++ to build professional games powered by DirectX, Unreal, or Cocos2d.

Components included by this workload

Component ID	Name	Version	Dependency Type
Microsoft.VisualStudio.Component.IntelliCode	IntelliCode	0.1	Required
Microsoft.VisualStudio.Component.VC.CoreRedist	C++ core features	16.0.28625.61	Required
Microsoft.VisualStudio.Component.VC.Redist.14.Latest	C++ 2019 Redistributable Update	16.4.29429.68	Required
Microsoft.VisualStudio.Component.VC.Tools.x86.x64	MSVC v142 - VS 2019 C++ x64/x86 build tools (v14.24)	16.4.29409.204	Required
Microsoft.VisualStudio.Component.Windows10SDK	Windows Universal C Runtime	16.4.29409.204	Required
Microsoft.VisualStudio.Component.Graphics.Tools	Graphics debugger and GPU profiler for DirectX	16.0.28625.61	Recommended
Microsoft.VisualStudio.Component.VC.ASAN	C++ AddressSanitizer (Experimental)	16.4.29429.68	Recommended
Microsoft.VisualStudio.Component.VC.DiagnosticTools	C++ profiling tools	16.4.29429.68	Recommended
Microsoft.VisualStudio.Component.Windows10SDK.18362	Windows 10 SDK (10.0.18362.0)	16.1.28829.92	Recommended
Component.Android.NDK.R16B	Android NDK (R16B)	16.4.29519.181	Optional

Component ID	Name	Version	Dependency Type
Component.Android.SDK25.Private	Android SDK setup (API level 25) (local install for Mobile development with C++)	16.0.28625.61	Optional
Component.Ant	Apache Ant (1.9.3)	1.9.3.8	Optional
Component.Cocos	Cocos	16.0.28315.86	Optional
Component.Incredibuild	Incredibuild - Build Acceleration	16.0.28528.71	Optional
Component.IncredibuildMenu	IncredibuildMenu	1.5.0.10	Optional
Component.MDD.Android	C++ Android development tools	16.0.28517.75	Optional
Component.OpenJDK	OpenJDK (Microsoft distribution)	16.1.28811.260	Optional
Component.Unreal	Unreal Engine installer	16.1.28810.153	Optional
Component.Unreal.Android	Android IDE support for Unreal engine	16.1.28810.153	Optional
Microsoft.Net.Component.4.5.1.TargetingPack	.NET Framework 4.5.1 targeting pack	16.0.28517.75	Optional
Microsoft.Net.Component.4.5.2.TargetingPack	.NET Framework 4.5.2 targeting pack	16.0.28517.75	Optional
Microsoft.Net.Component.4.5.TargetingPack	.NET Framework 4.5 targeting pack	16.0.28517.75	Optional
Microsoft.Net.Component.4.6.2.TargetingPack	.NET Framework 4.6.2 targeting pack	16.0.28517.75	Optional
Microsoft.Net.Component.4.6.TargetingPack	.NET Framework 4.6 targeting pack	16.0.28517.75	Optional
Microsoft.Net.Component.4.7.2.TargetingPack	.NET Framework 4.7.2 targeting pack	16.0.28517.75	Optional
Microsoft.Net.Component.4.8.SDK	.NET Framework 4.8 SDK	16.4.29313.120	Optional
Microsoft.Net.Component.4.TargetingPack	.NET Framework 4 targeting pack	16.0.28517.75	Optional
Microsoft.Net.ComponentGroup.DevelopmentPrerequisites	.NET Framework 4.7.2 development tools	16.3.29207.166	Optional

Component ID	Name	Version	Dependency Type
Microsoft.Net.ComponentGroup.TargetingPacks.Common	.NET Framework 4 – 4.6 development tools	16.0.28516.191	Optional
Microsoft.VisualStudio.Component.NuGet.BuildTools	NuGet targets and build tasks	16.1.28829.92	Optional
Microsoft.VisualStudio.Component.Roslyn.Compiler	C# and Visual Basic Roslyn compilers	16.0.28714.129	Optional
Microsoft.VisualStudio.Component.Roslyn.LanguageServices	C# and Visual Basic	16.4.29429.68	Optional
Microsoft.VisualStudio.Component.Windows10SDK.16299	Windows 10 SDK (10.0.16299.0)	16.0.28517.75	Optional
Microsoft.VisualStudio.Component.Windows10SDK.17134	Windows 10 SDK (10.0.17134.0)	16.0.28517.75	Optional
Microsoft.VisualStudio.Component.Windows10SDK.17763	Windows 10 SDK (10.0.17763.0)	16.0.28517.75	Optional

Mobile development with C++

ID: Microsoft.VisualStudio.Workload.NativeMobile

Description: Build cross-platform applications for iOS, Android or Windows using C++.

Components included by this workload

Component ID	Name	Version	Dependency Type
Component.Android.SDK25.Private	Android SDK setup (API level 25) (local install for Mobile development with C++)	16.0.28625.61	Required
Component.OpenJDK	OpenJDK (Microsoft distribution)	16.1.28811.260	Required
Microsoft.VisualStudio.Component.IntelliCode	IntelliCode	0.1	Required
Microsoft.VisualStudio.Component.VC.Corelde	C++ core features	16.0.28625.61	Required
Component.Android.NDK.R16B	Android NDK (R16B)	16.4.29519.181	Recommended
Component.Ant	Apache Ant (1.9.3)	1.9.3.8	Recommended

COMPONENT ID	NAME	VERSION	DEPENDENCY TYPE
Component.MDD.Android	C++ Android development tools	16.0.28517.75	Recommended
Component.Android.NDK.R16B_3264	Android NDK (R16B) (32bit)	16.4.29519.181	Optional
Component.Google.Android.Emulator.API25.Private	Google Android Emulator (API Level 25) (local install)	16.1.28810.153	Optional
Component.HAXM.Private	Intel Hardware Accelerated Execution Manager (HAXM) (local install)	16.0.28528.71	Optional
Component.Incredibuild	Incredibuild - Build Acceleration	16.0.28528.71	Optional
Component.IncredibuildMenu	IncredibuildMenu	1.5.0.10	Optional
Component.MDD.iOS	C++ iOS development tools	16.0.28517.75	Optional

.NET Core cross-platform development

ID: Microsoft.VisualStudio.Workload.NetCoreTools

Description: Build cross-platform applications using .NET Core, ASP.NET Core, HTML/JavaScript, and Containers including Docker support.

Components included by this workload

COMPONENT ID	NAME	VERSION	DEPENDENCY TYPE
Component.Microsoft.VisualStudio.RazorExtension	Razor Language Services	16.0.28714.129	Required
Component.Microsoft.Web.LibraryManager	Library Manager	16.0.28315.86	Required
Microsoft.Component_MSBUILD	MSBuild	16.4.29429.68	Required
Microsoft.Net.Component.4.5.2.TargetingPack	.NET Framework 4.5.2 targeting pack	16.0.28517.75	Required
Microsoft.Net.Component.4.5.TargetingPack	.NET Framework 4.5 targeting pack	16.0.28517.75	Required
Microsoft.Net.Component.4.6.1.TargetingPack	.NET Framework 4.6.1 targeting pack	16.0.28517.75	Required
Microsoft.Net.Component.4.7.2.TargetingPack	.NET Framework 4.7.2 targeting pack	16.0.28517.75	Required

Component ID	Name	Version	Dependency Type
Microsoft.Net.Component.4.8.SDK	.NET Framework 4.8 SDK	16.4.29313.120	Required
Microsoft.Net.ComponentGroup.DevelopmentPrerequisites	.NET Framework 4.7.2 development tools	16.3.29207.166	Required
Microsoft.NetCore.Component.DevelopmentTools	.NET Core development tools	16.4.29511.114	Required
Microsoft.NetCore.Component.SDK	.NET Core 3.1 SDK	16.4.29519.181	Required
Microsoft.NetCore.Component.Web	.NET Core development tools	16.4.29511.114	Required
Microsoft.VisualStudio.Component.Common.Azure.Tools	Connectivity and publishing tools	16.4.29409.204	Required
Microsoft.VisualStudio.Component.DockerTools	Container development tools	16.4.29409.204	Required
Microsoft.VisualStudio.Component.FSharp	F# language support	16.0.28315.86	Required
Microsoft.VisualStudio.Component.FSharp.WebTemplates	F# language support for web projects	16.3.29207.166	Required
Microsoft.VisualStudio.Component.IISExpress	IIS Express	16.0.28315.86	Required
Microsoft.VisualStudio.Component.IntelliCode	IntelliCode	0.1	Required
Microsoft.VisualStudio.Component.JavaScript.Diagnostics	JavaScript diagnostics	16.0.28517.75	Required
Microsoft.VisualStudio.Component.JavaScript.TypeScript	JavaScript and TypeScript language support	16.4.29409.204	Required
Microsoft.VisualStudio.Component.ManagedDesktop.Core	Managed Desktop Workload Core	16.4.29318.151	Required
Microsoft.VisualStudio.Component.MSODBC.SQL	SQL Server ODBC Driver	16.0.28625.61	Required
Microsoft.VisualStudio.Component.MSSQL.CMDLnUtils	SQL Server Command Line Utilities	16.0.28707.177	Required
Microsoft.VisualStudio.Component.NuGet	NuGet package manager	16.1.28829.92	Required

Component ID	Name	Version	Dependency Type
Microsoft.VisualStudio.Component.Roslyn.Compiler	C# and Visual Basic Roslyn compilers	16.0.28714.129	Required
Microsoft.VisualStudio.Component.Roslyn.LanguageServices	C# and Visual Basic	16.4.29429.68	Required
Microsoft.VisualStudio.Component.SQL.ADAL	SQL ADAL runtime	16.0.28517.75	Required
Microsoft.VisualStudio.Component.SQL.CLR	CLR data types for SQL Server	16.0.28315.86	Required
Microsoft.VisualStudio.Component.SQL.DataSources	Data sources for SQL Server support	16.0.28315.86	Required
Microsoft.VisualStudio.Component.SQL.LocalDB.Runtime	SQL Server Express 2016 LocalDB	16.0.28625.61	Required
Microsoft.VisualStudio.Component.SQL.SSDT	SQL Server Data Tools	16.3.29207.166	Required
Microsoft.VisualStudio.Component.TextTemplating	Text Template Transformation	16.0.28625.61	Required
Microsoft.VisualStudio.Component.TypeScript.3.7	TypeScript 3.7 SDK	16.0.29429.68	Required
Microsoft.VisualStudio.ComponentGroup.Web	ASP.NET and web development tools prerequisites	16.4.29318.151	Required
Microsoft.VisualStudio.ComponentGroup.WebToolsExtensions	ASP.NET and web development	16.0.28621.142	Required
Component.Microsoft.VisualStudio.LiveShare	Live Share	1.0.1076	Recommended
Component.Microsoft.VisualStudio.Web.AzureFunctions	Azure WebJobs Tools	16.0.28714.129	Recommended
Microsoft.Net.Core.Component.SDK.2.1	.NET Core 2.1 LTS Runtime	16.4.29519.181	Recommended
Microsoft.VisualStudio.Component.AppInsights.Tools	Developer Analytics tools	16.4.29429.68	Recommended
Microsoft.VisualStudio.Component.Azure.AuthoringTools	Azure Authoring Tools	16.0.28625.61	Recommended
Microsoft.VisualStudio.Component.Azure.ClientLibs	Azure libraries for .NET	16.0.28315.86	Recommended

COMPONENT ID	NAME	VERSION	DEPENDENCY TYPE
Microsoft.VisualStudio.Component.Azure.Compute Emulator	Azure Compute Emulator	16.1.28810.153	Recommended
Microsoft.VisualStudio.Component.Azure.Storage.Emulator	Azure Storage Emulator	16.4.29313.120	Recommended
Microsoft.VisualStudio.Component.CloudExplorer	Cloud Explorer	16.0.28625.61	Recommended
Microsoft.VisualStudio.Component.DiagnosticTools	.NET profiling tools	16.4.29429.68	Recommended
Microsoft.VisualStudio.Component.Web	ASP.NET and web development tools	16.0.28517.75	Recommended
Microsoft.VisualStudio.Component.WebDeploy	Web Deploy	16.0.28517.75	Recommended
Microsoft.VisualStudio.ComponentGroup.AzureFunctions	Azure WebJobs Tools	16.0.28621.142	Recommended
Microsoft.VisualStudio.ComponentGroup.Web.CloudTools	Cloud tools for web development	16.2.29003.222	Recommended
Microsoft.VisualStudio.Component.Windows10SDK.18362	Windows 10 SDK (10.0.18362.0)	16.1.28829.92	Optional
Microsoft.VisualStudio.ComponentGroup.IISDevelopment	Development time IIS support	16.0.28315.86	Optional
Microsoft.VisualStudio.ComponentGroup.MSIX.Packaging	MSIX Packaging Tools	16.4.29409.204	Optional

Mobile development with .NET

ID: Microsoft.VisualStudio.Workload.NetCrossPlat

Description: Build cross-platform applications for iOS, Android or Windows using Xamarin.

Components included by this workload

COMPONENT ID	NAME	VERSION	DEPENDENCY TYPE
Component.OpenJDK	OpenJDK (Microsoft distribution)	16.1.28811.260	Required
Component.Xamarin	Xamarin	16.4.29409.204	Required
Component.Xamarin.RemotedSimulator	Xamarin Remoted Simulator	16.0.28315.86	Required

Component ID	Name	Version	Dependency Type
Microsoft.Component.MSBuild	MSBuild	16.4.29429.68	Required
Microsoft.Net.Component.4.6.1.TargetingPack	.NET Framework 4.6.1 targeting pack	16.0.28517.75	Required
Microsoft.Net.Component.4.7.2.TargetingPack	.NET Framework 4.7.2 targeting pack	16.0.28517.75	Required
Microsoft.Net.Component.4.8.SDK	.NET Framework 4.8 SDK	16.4.29313.120	Required
Microsoft.Net.ComponentGroup.DevelopmentPrerequisites	.NET Framework 4.7.2 development tools	16.3.29207.166	Required
Microsoft.NetCore.Component.DevelopmentTools	.NET Core development tools	16.4.29511.114	Required
Microsoft.NetCore.Component.SDK	.NET Core 3.1 SDK	16.4.29519.181	Required
Microsoft.VisualStudio.Component.FSharp	F# language support	16.0.28315.86	Required
Microsoft.VisualStudio.Component.IntelliCode	IntelliCode	0.1	Required
Microsoft.VisualStudio.Component.Merq	Common Xamarin internal tools	16.2.29012.281	Required
Microsoft.VisualStudio.Component.MonoDebugger	Mono debugger	16.0.28517.75	Required
Microsoft.VisualStudio.Component.NuGet	NuGet package manager	16.1.28829.92	Required
Microsoft.VisualStudio.Component.Roslyn.Compiler	C# and Visual Basic Roslyn compilers	16.0.28714.129	Required
Microsoft.VisualStudio.Component.Roslyn.LanguageServices	C# and Visual Basic	16.4.29429.68	Required
Microsoft.VisualStudio.ComponentGroup.WebToolsExtensions.TemplateEngine	ASP.NET templating engine	16.0.28315.86	Required
Component.Android.SDK28	Android SDK setup (API level 28)	16.2.29003.222	Recommended

ASP.NET and web development

ID: Microsoft.VisualStudio.Workload.NetWeb

Description: Build web applications using ASP.NET Core, ASP.NET, HTML/JavaScript, and Containers including Docker support.

Components included by this workload

Component ID	Name	Version	Dependency Type
Component.Microsoft.VisualStudio.RazorExtension	Razor Language Services	16.0.28714.129	Required
Component.Microsoft.Web.LibraryManager	Library Manager	16.0.28315.86	Required
Microsoft.Component.MSBuild	MSBuild	16.4.29429.68	Required
Microsoft.Net.Component.4.5.2.TargetingPack	.NET Framework 4.5.2 targeting pack	16.0.28517.75	Required
Microsoft.Net.Component.4.5.TargetingPack	.NET Framework 4.5 targeting pack	16.0.28517.75	Required
Microsoft.Net.Component.4.6.1.TargetingPack	.NET Framework 4.6.1 targeting pack	16.0.28517.75	Required
Microsoft.Net.Component.4.7.2.TargetingPack	.NET Framework 4.7.2 targeting pack	16.0.28517.75	Required
Microsoft.Net.Component.4.8.SDK	.NET Framework 4.8 SDK	16.4.29313.120	Required
Microsoft.Net.ComponentGroup.DevelopmentPrerequisites	.NET Framework 4.7.2 development tools	16.3.29207.166	Required
Microsoft.NetCore.Component.DevelopmentTools	.NET Core development tools	16.4.29511.114	Required
Microsoft.NetCore.Component.SDK	.NET Core 3.1 SDK	16.4.29519.181	Required
Microsoft.NetCore.Component.Web	.NET Core development tools	16.4.29511.114	Required
Microsoft.VisualStudio.Component.Common.Azure.Tools	Connectivity and publishing tools	16.4.29409.204	Required
Microsoft.VisualStudio.Component.DockerTools	Container development tools	16.4.29409.204	Required
Microsoft.VisualStudio.Component.FSharp	F# language support	16.0.28315.86	Required
Microsoft.VisualStudio.Component.FSharp.WebTemplates	F# language support for web projects	16.3.29207.166	Required

Component ID	Name	Version	Dependency Type
Microsoft.VisualStudio.Component.IISExpress	IIS Express	16.0.28315.86	Required
Microsoft.VisualStudio.Component.IntelliCode	IntelliCode	0.1	Required
Microsoft.VisualStudio.Component.JavaScript.Diagnostics	JavaScript diagnostics	16.0.28517.75	Required
Microsoft.VisualStudio.Component.JavaScript.TypeScript	JavaScript and TypeScript language support	16.4.29409.204	Required
Microsoft.VisualStudio.Component.ManagedDesktop.Core	Managed Desktop Workload Core	16.4.29318.151	Required
Microsoft.VisualStudio.Component.MSODBC.SQL	SQL Server ODBC Driver	16.0.28625.61	Required
Microsoft.VisualStudio.Component.MSSQL.CMDLnUtils	SQL Server Command Line Utilities	16.0.28707.177	Required
Microsoft.VisualStudio.Component.NuGet	NuGet package manager	16.1.28829.92	Required
Microsoft.VisualStudio.Component.Roslyn.Compiler	C# and Visual Basic Roslyn compilers	16.0.28714.129	Required
Microsoft.VisualStudio.Component.Roslyn.LanguageServices	C# and Visual Basic	16.4.29429.68	Required
Microsoft.VisualStudio.Component.SQL.ADAL	SQL ADAL runtime	16.0.28517.75	Required
Microsoft.VisualStudio.Component.SQL.CLR	CLR data types for SQL Server	16.0.28315.86	Required
Microsoft.VisualStudio.Component.SQL.DataSources	Data sources for SQL Server support	16.0.28315.86	Required
Microsoft.VisualStudio.Component.SQL.LocalDB.Runtime	SQL Server Express 2016 LocalDB	16.0.28625.61	Required
Microsoft.VisualStudio.Component.SQL.SSDT	SQL Server Data Tools	16.3.29207.166	Required
Microsoft.VisualStudio.Component.TextTemplating	Text Template Transformation	16.0.28625.61	Required
Microsoft.VisualStudio.Component.TypeScript.3.7	TypeScript 3.7 SDK	16.0.29429.68	Required

Component ID	Name	Version	Dependency Type
Microsoft.VisualStudio.Component.Web	ASP.NET and web development tools	16.0.28517.75	Required
Microsoft.VisualStudio.ComponentGroup.Web	ASP.NET and web development tools prerequisites	16.4.29318.151	Required
Microsoft.VisualStudio.ComponentGroup.WebToolsExtensions	ASP.NET and web development	16.0.28621.142	Required
Component.Microsoft.VisualStudio.LiveShare	Live Share	1.0.1076	Recommended
Component.Microsoft.VisualStudio.Web.AzureFunctions	Azure WebJobs Tools	16.0.28714.129	Recommended
Microsoft.Net.Component.4.5.1.TargetingPack	.NET Framework 4.5.1 targeting pack	16.0.28517.75	Recommended
Microsoft.Net.Component.4.6.TargetingPack	.NET Framework 4.6 targeting pack	16.0.28517.75	Recommended
Microsoft.Net.Component.4.TargetingPack	.NET Framework 4 targeting pack	16.0.28517.75	Recommended
Microsoft.Net.ComponentGroup.TargetingPacks.Common	.NET Framework 4 – 4.6 development tools	16.0.28516.191	Recommended
Microsoft.Net.Core.Component.SDK.2.1	.NET Core 2.1 LTS Runtime	16.4.29519.181	Recommended
Microsoft.VisualStudio.Component.AppInsights.Tools	Developer Analytics tools	16.4.29429.68	Recommended
Microsoft.VisualStudio.Component.AspNet45	Advanced ASP.NET features	16.0.28315.86	Recommended
Microsoft.VisualStudio.Component.Azure.AuthoringTools	Azure Authoring Tools	16.0.28625.61	Recommended
Microsoft.VisualStudio.Component.Azure.ClientLibs	Azure libraries for .NET	16.0.28315.86	Recommended
Microsoft.VisualStudio.Component.Azure.Compute.Emulator	Azure Compute Emulator	16.1.28810.153	Recommended
Microsoft.VisualStudio.Component.Azure.Storage.Emulator	Azure Storage Emulator	16.4.29313.120	Recommended

Component ID	Name	Version	Dependency Type
Microsoft.VisualStudio.Component.CloudExplorer	Cloud Explorer	16.0.28625.61	Recommended
Microsoft.VisualStudio.Component.DiagnosticTools	.NET profiling tools	16.4.29429.68	Recommended
Microsoft.VisualStudio.Component.EntityFramework	Entity Framework 6 tools	16.0.28315.86	Recommended
Microsoft.VisualStudio.Component.WebDeploy	Web Deploy	16.0.28517.75	Recommended
Microsoft.VisualStudio.ComponentGroup.AzureFunctions	Azure WebJobs Tools	16.0.28621.142	Recommended
Microsoft.VisualStudio.ComponentGroup.Web.CloudTools	Cloud tools for web development	16.2.29003.222	Recommended
Microsoft.Net.Component.4.6.2.TargetingPack	.NET Framework 4.6.2 targeting pack	16.0.28517.75	Optional
Microsoft.Net.Component.4.7.1.TargetingPack	.NET Framework 4.7.1 targeting pack	16.0.28517.75	Optional
Microsoft.Net.Component.4.7.TargetingPack	.NET Framework 4.7 targeting pack	16.0.28517.75	Optional
Microsoft.Net.Component.4.8.TargetingPack	.NET Framework 4.8 targeting pack	16.4.29313.120	Optional
Microsoft.Net.ComponentGroup.4.6.1.DeveloperTools	.NET Framework 4.6.1 development tools	16.3.29207.166	Optional
Microsoft.Net.ComponentGroup.4.6.2.DeveloperTools	.NET Framework 4.6.2 development tools	16.3.29207.166	Optional
Microsoft.Net.ComponentGroup.4.7.1.DeveloperTools	.NET Framework 4.7.1 development tools	16.3.29207.166	Optional
Microsoft.Net.ComponentGroup.4.7.DeveloperTools	.NET Framework 4.7 development tools	16.3.29207.166	Optional
Microsoft.Net.ComponentGroup.4.8.DeveloperTools	.NET Framework 4.8 development tools	16.4.29318.151	Optional
Microsoft.VisualStudio.Component.Wcf.Tooling	Windows Communication Foundation	16.0.28625.61	Optional
Microsoft.VisualStudio.ComponentGroup.AdditionalWebProjectTemplates	Additional project templates (previous versions)	16.0.28621.142	Optional
Microsoft.VisualStudio.ComponentGroup.IISDevelopment	Development time IIS support	16.0.28315.86	Optional

Node.js development

ID: Microsoft.VisualStudio.Workload.Node

Description: Build scalable network applications using Node.js, an asynchronous event-driven JavaScript runtime.

Components included by this workload

Component ID	Name	Version	Dependency Type
Microsoft.VisualStudio.Component.JavaScript.Diagnostics	JavaScript diagnostics	16.0.28517.75	Required
Microsoft.VisualStudio.Component.JavaScript.TypeScript	JavaScript and TypeScript language support	16.4.29409.204	Required
Microsoft.VisualStudio.Component.Node.Tools	Node.js development tools	16.4.29429.68	Required
Microsoft.VisualStudio.Component.TypeScript.3.7	TypeScript 3.7 SDK	16.0.29429.68	Required
Microsoft.VisualStudio.ComponentGroup.WebToolsExtensions	ASP.NET and web development	16.0.28621.142	Required
Component.Microsoft.VisualStudio.LiveShare	Live Share	1.0.1076	Recommended
Microsoft.VisualStudio.Component.IntelliCode	IntelliCode	0.1	Recommended
Microsoft.VisualStudio.Component.WebDeploy	Web Deploy	16.0.28517.75	Recommended
Microsoft.VisualStudio.Component.AppInsights.Tools	Developer Analytics tools	16.4.29429.68	Optional
Microsoft.VisualStudio.Component.Common.Azure.Tools	Connectivity and publishing tools	16.4.29409.204	Optional
Microsoft.VisualStudio.Component.VC.Corelde	C++ core features	16.0.28625.61	Optional
Microsoft.VisualStudio.Component.VC.Tools.x86.x64	MSVC v142 - VS 2019 C++ x64/x86 build tools (v14.24)	16.4.29409.204	Optional

Office/SharePoint development

ID: Microsoft.VisualStudio.Workload.Office

Description: Create Office and SharePoint add-ins, SharePoint solutions, and VSTO add-ins using C#, VB, and JavaScript.

Components included by this workload

Component ID	Name	Version	Dependency Type
Component.Microsoft.VisualStudio.RazorExtension	Razor Language Services	16.0.28714.129	Required
Component.Microsoft.Web.LibraryManager	Library Manager	16.0.28315.86	Required
Microsoft.Component.MSBuild	MSBuild	16.4.29429.68	Required
Microsoft.Net.Component.4.5.2.TargetingPack	.NET Framework 4.5.2 targeting pack	16.0.28517.75	Required
Microsoft.Net.Component.4.5.TargetingPack	.NET Framework 4.5 targeting pack	16.0.28517.75	Required
Microsoft.Net.Component.4.6.1.TargetingPack	.NET Framework 4.6.1 targeting pack	16.0.28517.75	Required
Microsoft.Net.Component.4.7.2.TargetingPack	.NET Framework 4.7.2 targeting pack	16.0.28517.75	Required
Microsoft.Net.Component.4.8.SDK	.NET Framework 4.8 SDK	16.4.29313.120	Required
Microsoft.Net.Component.4.TargetingPack	.NET Framework 4 targeting pack	16.0.28517.75	Required
Microsoft.Net.ComponentGroup.DevelopmentPrerequisites	.NET Framework 4.7.2 development tools	16.3.29207.166	Required
Microsoft.NetCore.Component.SDK	.NET Core 3.1 SDK	16.4.29519.181	Required
Microsoft.VisualStudio.Component.AppInsights.Tools	Developer Analytics tools	16.4.29429.68	Required
Microsoft.VisualStudio.Component.Common.Azure.Tools	Connectivity and publishing tools	16.4.29409.204	Required
Microsoft.VisualStudio.Component.DockerTools	Container development tools	16.4.29409.204	Required
Microsoft.VisualStudio.Component.IISExpress	IIS Express	16.0.28315.86	Required
Microsoft.VisualStudio.Component.IntelliCode	IntelliCode	0.1	Required
Microsoft.VisualStudio.Component.JavaScript.Diagnostics	JavaScript diagnostics	16.0.28517.75	Required
Microsoft.VisualStudio.Component.JavaScript.TypeScript	JavaScript and TypeScript language support	16.4.29409.204	Required

Component ID	Name	Version	Dependency Type
Microsoft.VisualStudio.Component.ManagedDesktop.Core	Managed Desktop Workload Core	16.4.29318.151	Required
Microsoft.VisualStudio.Component.ManagedDesktop.Prerequisites	.NET desktop development tools	16.4.29429.68	Required
Microsoft.VisualStudio.Component.MSODBC.SQL	SQL Server ODBC Driver	16.0.28625.61	Required
Microsoft.VisualStudio.Component.MSSQL.CMDLnUtils	SQL Server Command Line Utilities	16.0.28707.177	Required
Microsoft.VisualStudio.Component.NuGet	NuGet package manager	16.1.28829.92	Required
Microsoft.VisualStudio.Component.Roslyn.Compiler	C# and Visual Basic Roslyn compilers	16.0.28714.129	Required
Microsoft.VisualStudio.Component.Roslyn.LanguageServices	C# and Visual Basic	16.4.29429.68	Required
Microsoft.VisualStudio.Component.Sharepoint.Tools	Office Developer Tools for Visual Studio	16.4.29409.204	Required
Microsoft.VisualStudio.Component.SQL.ADAL	SQL ADAL runtime	16.0.28517.75	Required
Microsoft.VisualStudio.Component.SQL.CLR	CLR data types for SQL Server	16.0.28315.86	Required
Microsoft.VisualStudio.Component.SQL.DataSources	Data sources for SQL Server support	16.0.28315.86	Required
Microsoft.VisualStudio.Component.SQL.LocalDB.Runtime	SQL Server Express 2016 LocalDB	16.0.28625.61	Required
Microsoft.VisualStudio.Component.SQL.SSDT	SQL Server Data Tools	16.3.29207.166	Required
Microsoft.VisualStudio.Component.TextTemplating	Text Template Transformation	16.0.28625.61	Required
Microsoft.VisualStudio.Component.TypeScript.3.7	TypeScript 3.7 SDK	16.0.29429.68	Required
Microsoft.VisualStudio.Component.Wcf.Tooling	Windows Communication Foundation	16.0.28625.61	Required
Microsoft.VisualStudio.Component.Web	ASP.NET and web development tools	16.0.28517.75	Required

Component ID	Name	Version	Dependency Type
Microsoft.VisualStudio.Component.Workflow	Windows Workflow Foundation	16.0.28315.86	Required
Microsoft.VisualStudio.ComponentGroup.Web	ASP.NET and web development tools prerequisites	16.4.29318.151	Required
Microsoft.VisualStudio.ComponentGroup.WebToolsExtensions	ASP.NET and web development	16.0.28621.142	Required
Microsoft.VisualStudio.Component.TeamOffice	Visual Studio Tools for Office (VSTO)	16.4.29409.204	Recommended
Microsoft.VisualStudio.Component.WebDeploy	Web Deploy	16.0.28517.75	Recommended
Microsoft.Net.Component.4.6.2.TargetingPack	.NET Framework 4.6.2 targeting pack	16.0.28517.75	Optional
Microsoft.Net.Component.4.7.1.TargetingPack	.NET Framework 4.7.1 targeting pack	16.0.28517.75	Optional
Microsoft.Net.Component.4.7.TargetingPack	.NET Framework 4.7 targeting pack	16.0.28517.75	Optional
Microsoft.Net.Component.4.8.TargetingPack	.NET Framework 4.8 targeting pack	16.4.29313.120	Optional
Microsoft.Net.ComponentGroup.4.6.1.DeveloperTools	.NET Framework 4.6.1 development tools	16.3.29207.166	Optional
Microsoft.Net.ComponentGroup.4.6.2.DeveloperTools	.NET Framework 4.6.2 development tools	16.3.29207.166	Optional
Microsoft.Net.ComponentGroup.4.7.1.DeveloperTools	.NET Framework 4.7.1 development tools	16.3.29207.166	Optional
Microsoft.Net.ComponentGroup.4.7.DeveloperTools	.NET Framework 4.7 development tools	16.3.29207.166	Optional
Microsoft.Net.ComponentGroup.4.8.DeveloperTools	.NET Framework 4.8 development tools	16.4.29318.151	Optional
Microsoft.VisualStudio.ComponentGroup.Sharepoint.WIF	Windows Identity Foundation 3.5	16.0.28621.142	Optional

Python development

ID: Microsoft.VisualStudio.Workload.Python

Description: Editing, debugging, interactive development and source control for Python.

Components included by this workload

Component ID	Name	Version	Dependency Type
Microsoft.Component.PythonTools	Python language support	16.4.29429.68	Required
Component.CPython3.x64	Python 3 64-bit (3.7.5)	3.7.5	Recommended
Component.Microsoft.VisualStudio.LiveShare	Live Share	1.0.1076	Recommended
Microsoft.Component.PythonTools.Minicondax64	Python miniconda	16.2.29003.222	Recommended
Microsoft.Component.PythonTools.Web	Python web support	16.0.28517.75	Recommended
Microsoft.VisualStudio.Component.Common.Azure.Tools	Connectivity and publishing tools	16.4.29409.204	Recommended
Microsoft.VisualStudio.Component.JavaScript.TypeScript	JavaScript and TypeScript language support	16.4.29409.204	Recommended
Microsoft.VisualStudio.Component.TypeScript.3.7	TypeScript 3.7 SDK	16.0.29429.68	Recommended
Microsoft.VisualStudio.Component.WebDeploy	Web Deploy	16.0.28517.75	Recommended
Microsoft.VisualStudio.ComponentGroup.WebToolsExtensions	ASP.NET and web development	16.0.28621.142	Recommended
Component.CPython2.x64	Python 2 64-bit (2.7.16)	2.7.16	Optional
Component.CPython2.x86	Python 2 32-bit (2.7.16)	2.7.16	Optional
Component.CPython3.x86	Python 3 32-bit (3.7.5)	3.7.5	Optional
Component.Microsoft.VisualStudio.RazorExtension	Razor Language Services	16.0.28714.129	Optional
Component.Microsoft.Web.LibraryManager	Library Manager	16.0.28315.86	Optional
Microsoft.Component.MSBuild	MSBuild	16.4.29429.68	Optional
Microsoft.ComponentGroup.PythonTools.NativeDevelopment	Python native development tools	16.2.29020.229	Optional
Microsoft.Net.Component.4.5.2.TargetingPack	.NET Framework 4.5.2 targeting pack	16.0.28517.75	Optional

Component ID	Name	Version	Dependency Type
Microsoft.Net.Component.4.5.TargetingPack	.NET Framework 4.5 targeting pack	16.0.28517.75	Optional
Microsoft.Net.Component.4.7.2.TargetingPack	.NET Framework 4.7.2 targeting pack	16.0.28517.75	Optional
Microsoft.Net.Component.4.8.SDK	.NET Framework 4.8 SDK	16.4.29313.120	Optional
Microsoft.Net.ComponentGroup.DevelopmentPrerequisites	.NET Framework 4.7.2 development tools	16.3.29207.166	Optional
Microsoft.NetCore.Component.SDK	.NET Core 3.1 SDK	16.4.29519.181	Optional
Microsoft.VisualStudio.Component.Azure.AuthoringTools	Azure Authoring Tools	16.0.28625.61	Optional
Microsoft.VisualStudio.Component.Azure.ClientLibs	Azure libraries for .NET	16.0.28315.86	Optional
Microsoft.VisualStudio.Component.Azure.Compute.Emulator	Azure Compute Emulator	16.1.28810.153	Optional
Microsoft.VisualStudio.Component.Azure.Storage.Emulator	Azure Storage Emulator	16.4.29313.120	Optional
Microsoft.VisualStudio.Component.Azure.Waverton	Azure Cloud Services core tools	16.4.29409.204	Optional
Microsoft.VisualStudio.Component.Azure.Waverton.BuildTools	Azure Cloud Services build tools	16.3.29207.166	Optional
Microsoft.VisualStudio.Component.DockerTools	Container development tools	16.4.29409.204	Optional
Microsoft.VisualStudio.Component.Graphics.Tools	Graphics debugger and GPU profiler for DirectX	16.0.28625.61	Optional
Microsoft.VisualStudio.Component.IISExpress	IIS Express	16.0.28315.86	Optional
Microsoft.VisualStudio.Component.JavaScript.Diagnostics	JavaScript diagnostics	16.0.28517.75	Optional
Microsoft.VisualStudio.Component.ManagedDesktop.Core	Managed Desktop Workload Core	16.4.29318.151	Optional

Component ID	Name	Version	Dependency Type
Microsoft.VisualStudio.Component.MSODBC.SQL	SQL Server ODBC Driver	16.0.28625.61	Optional
Microsoft.VisualStudio.Component.MSSQL.CMDLnUtils	SQL Server Command Line Utilities	16.0.28707.177	Optional
Microsoft.VisualStudio.Component.NuGet	NuGet package manager	16.1.28829.92	Optional
Microsoft.VisualStudio.Component.Roslyn.Compiler	C# and Visual Basic Roslyn compilers	16.0.28714.129	Optional
Microsoft.VisualStudio.Component.Roslyn.LanguageServices	C# and Visual Basic	16.4.29429.68	Optional
Microsoft.VisualStudio.Component.SQL.ADAL	SQL ADAL runtime	16.0.28517.75	Optional
Microsoft.VisualStudio.Component.SQL.CLR	CLR data types for SQL Server	16.0.28315.86	Optional
Microsoft.VisualStudio.Component.SQL.DataSources	Data sources for SQL Server support	16.0.28315.86	Optional
Microsoft.VisualStudio.Component.SQL.LocalDB.Runtime	SQL Server Express 2016 LocalDB	16.0.28625.61	Optional
Microsoft.VisualStudio.Component.SQL.SSDT	SQL Server Data Tools	16.3.29207.166	Optional
Microsoft.VisualStudio.Component.TextTemplating	Text Template Transformation	16.0.28625.61	Optional
Microsoft.VisualStudio.Component.VC.Coreide	C++ core features	16.0.28625.61	Optional
Microsoft.VisualStudio.Component.VC.DiagnosticTools	C++ profiling tools	16.4.29429.68	Optional
Microsoft.VisualStudio.Component.VC.Tools.x86.x64	MSVC v142 - VS 2019 C++ x64/x86 build tools (v14.24)	16.4.29409.204	Optional
Microsoft.VisualStudio.Component.Web	ASP.NET and web development tools	16.0.28517.75	Optional
Microsoft.VisualStudio.Component.Windows10SDK	Windows Universal C Runtime	16.4.29409.204	Optional
Microsoft.VisualStudio.Component.Windows10SDK.18362	Windows 10 SDK (10.0.18362.0)	16.1.28829.92	Optional

Component ID	Name	Version	Dependency Type
Microsoft.VisualStudio.ComponentGroup.Web	ASP.NET and web development tools prerequisites	16.4.29318.151	Optional

Universal Windows Platform development

ID: Microsoft.VisualStudio.Workload.Universal

Description: Create applications for the Universal Windows Platform with C#, VB, or optionally C++.

Components included by this workload

Component ID	Name	Version	Dependency Type
Microsoft.Component.NetFX.Native	.NET Native	16.4.29429.68	Required
Microsoft.ComponentGroup.Blend	Blend for Visual Studio	16.0.28315.86	Required
Microsoft.Net.Component.4.5.TargetingPack	.NET Framework 4.5 targeting pack	16.0.28517.75	Required
Microsoft.NetCore.Component.SDK	.NET Core 3.1 SDK	16.4.29519.181	Required
Microsoft.VisualStudio.Component.AppInsights.Tools	Developer Analytics tools	16.4.29429.68	Required
Microsoft.VisualStudio.Component.DiagnosticTools	.NET profiling tools	16.4.29429.68	Required
Microsoft.VisualStudio.Component.Graphics	Image and 3D model editors	16.0.28517.75	Required
Microsoft.VisualStudio.Component.IntelliCode	IntelliCode	0.1	Required
Microsoft.VisualStudio.Component.NuGet	NuGet package manager	16.1.28829.92	Required
Microsoft.VisualStudio.Component.Roslyn.Compiler	C# and Visual Basic Roslyn compilers	16.0.28714.129	Required
Microsoft.VisualStudio.Component.Roslyn.LanguageServices	C# and Visual Basic	16.4.29429.68	Required
Microsoft.VisualStudio.Component.SQL.CLR	CLR data types for SQL Server	16.0.28315.86	Required
Microsoft.VisualStudio.Component.Windows10SDK.18362	Windows 10 SDK (10.0.18362.0)	16.1.28829.92	Required

Component ID	Name	Version	Dependency Type
Microsoft.VisualStudio.ComponentGroup.MSIX.Packaging	MSIX Packaging Tools	16.4.29409.204	Required
Microsoft.VisualStudio.ComponentGroup.UWP.NetCoreAndStandard	.NET Native and .NET Standard	16.3.29102.218	Required
Microsoft.VisualStudio.ComponentGroup.UWP.Support	Universal Windows Platform tools	16.4.29409.204	Required
Microsoft.VisualStudio.ComponentGroup.UWP.Xamarin	Universal Windows Platform tools for Xamarin	16.4.29511.114	Required
Microsoft.Component.MSBuild	MSBuild	16.4.29429.68	Optional
Microsoft.Net.Component.4.8.SDK	.NET Framework 4.8 SDK	16.4.29313.120	Optional
Microsoft.VisualStudio.Component.Graphics.Tools	Graphics debugger and GPU profiler for DirectX	16.0.28625.61	Optional
Microsoft.VisualStudio.Component.TextTemplating	Text Template Transformation	16.0.28625.61	Optional
Microsoft.VisualStudio.Component.UWP.VC.ARM64	C++ Universal Windows Platform support for v142 build tools (ARM64)	16.3.29207.166	Optional
Microsoft.VisualStudio.Component.VC.CoreLde	C++ core features	16.0.28625.61	Optional
Microsoft.VisualStudio.Component.VC.Redist.14.Latest	C++ 2019 Redistributable Update	16.4.29429.68	Optional
Microsoft.VisualStudio.Component.VC.Tools.ARM	MSVC v142 - VS 2019 C++ ARM build tools (v14.24)	16.4.29409.204	Optional
Microsoft.VisualStudio.Component.VC.Tools.ARM64	MSVC v142 - VS 2019 C++ ARM64 build tools (v14.24)	16.4.29409.204	Optional
Microsoft.VisualStudio.Component.VC.Tools.x86.x64	MSVC v142 - VS 2019 C++ x64/x86 build tools (v14.24)	16.4.29409.204	Optional
Microsoft.VisualStudio.Component.VC.v141.ARM	MSVC v141 - VS 2017 C++ ARM build tools (v14.16)	16.2.29003.222	Optional
Microsoft.VisualStudio.Component.VC.v141.ARM64	MSVC v141 - VS 2017 C++ ARM64 build tools (v14.16)	16.1.28829.92	Optional
Microsoft.VisualStudio.Component.VC.v141.x86.x64	MSVC v141 - VS 2017 C++ x64/x86 build tools (v14.16)	16.1.28829.92	Optional

Component ID	Name	Version	Dependency Type
Microsoft.VisualStudio.Component.Windows10SDK.16299	Windows 10 SDK (10.0.16299.0)	16.0.28517.75	Optional
Microsoft.VisualStudio.Component.Windows10SDK.17134	Windows 10 SDK (10.0.17134.0)	16.0.28517.75	Optional
Microsoft.VisualStudio.Component.Windows10SDK.17763	Windows 10 SDK (10.0.17763.0)	16.0.28517.75	Optional
Microsoft.VisualStudio.Component.Windows10SDK.IoPOverUsb	USB Device Connectivity	16.4.29511.114	Optional
Microsoft.VisualStudio.ComponentGroup.NativeDesktop.Core	C++ core desktop features	16.2.29012.281	Optional
Microsoft.VisualStudio.ComponentGroup.UWP.VC	C++ (v142) Universal Windows Platform tools	16.3.29207.166	Optional
Microsoft.VisualStudio.ComponentGroup.UWP.VC.v141	C++ (v141) Universal Windows Platform tools	16.1.28810.153	Optional

Visual Studio extension development

ID: Microsoft.VisualStudio.Workload.VisualStudioExtension

Description: Create add-ons and extensions for Visual Studio, including new commands, code analyzers and tool windows.

Components included by this workload

Component ID	Name	Version	Dependency Type
Microsoft.Component.MSBuild	MSBuild	16.4.29429.68	Required
Microsoft.Net.Component.4.6.TargetingPack	.NET Framework 4.6 targeting pack	16.0.28517.75	Required
Microsoft.Net.Component.4.7.2.TargetingPack	.NET Framework 4.7.2 targeting pack	16.0.28517.75	Required
Microsoft.Net.Component.4.8.SDK	.NET Framework 4.8 SDK	16.4.29313.120	Required
Microsoft.Net.ComponentGroup.DevelopmentPrerequisites	.NET Framework 4.7.2 development tools	16.3.29207.166	Required

Component ID	Name	Version	Dependency Type
Microsoft.VisualStudio.Component.IntelliCode	IntelliCode	0.1	Required
Microsoft.VisualStudio.Component.NuGet	NuGet package manager	16.1.28829.92	Required
Microsoft.VisualStudio.Component.Roslyn.Compiler	C# and Visual Basic Roslyn compilers	16.0.28714.129	Required
Microsoft.VisualStudio.Component.Roslyn.LanguageServices	C# and Visual Basic	16.4.29429.68	Required
Microsoft.VisualStudio.Component.VSSDK	Visual Studio SDK	16.0.28315.86	Required
Microsoft.VisualStudio.ComponentGroup.VisualStudioExtension.Prerequisites	Visual Studio extension development prerequisites	16.4.29318.151	Required
Microsoft.VisualStudio.Component.DiagnosticTools	.NET profiling tools	16.4.29429.68	Recommended
Microsoft.VisualStudio.Component.TextTemplating	Text Template Transformation	16.0.28625.61	Recommended
Microsoft.Component.CodeAnalysis.SDK	.NET Compiler Platform SDK	16.2.29003.222	Optional
Microsoft.VisualStudio.Component.AppInsights.Tools	Developer Analytics tools	16.4.29429.68	Optional
Microsoft.VisualStudio.Component.DslTools	Modeling SDK	16.0.28315.86	Optional

Unaffiliated components

These are components that are not included with any workload, but may be selected as an individual component.

Component ID	Name	Version
Component.GitHub.VisualStudio	GitHub extension for Visual Studio	2.5.9.5485
Component.Xamarin.Inspector	Xamarin Inspector	16.0.28315.86
Component.Xamarin.Profiler	Xamarin Profiler	16.0.28315.86
Component.Xamarin.Workbooks	Xamarin Workbooks	16.0.28315.86
Microsoft.Component.ClickOnce	ClickOnce Publishing	16.4.29409.204
Microsoft.Component.HelpViewer	Help Viewer	16.0.28625.61

Component ID	Name	Version
Microsoft.Net.Component.4.6.1.SDK	.NET Framework 4.6.1 SDK	16.4.29409.204
Microsoft.Net.Component.4.6.2.SDK	.NET Framework 4.6.2 SDK	16.4.29409.204
Microsoft.Net.Component.4.7.1.SDK	.NET Framework 4.7.1 SDK	16.4.29409.204
Microsoft.Net.Component.4.7.2.SDK	.NET Framework 4.7.2 SDK	16.4.29409.204
Microsoft.Net.Component.4.7.SDK	.NET Framework 4.7 SDK	16.4.29409.204
Microsoft.Net.Core.Component.SDK.2.2	.NET Core 2.2 Runtime	16.4.29519.181
Microsoft.Net.Core.Component.SDK.3.0	.NET Core 3.0 Runtime	16.4.29519.181
Microsoft.NetCore.ComponentGroup.DevelopmentTools.2.1	Development Tools plus .NET Core 2.1	16.3.29207.166
Microsoft.NetCore.ComponentGroup.Web.2.1	Web Development Tools plus .NET Core 2.1	16.3.29207.166
Microsoft.VisualStudio.Component.AzureDevOps.OfficeIntegration	Azure DevOps Office Integration	16.0.28625.61
Microsoft.VisualStudio.Component.ClassDesigner	Class Designer	16.0.28528.71
Microsoft.VisualStudio.Component.DependencyValidation.Community	Dependency Validation	16.0.28517.75
Microsoft.VisualStudio.Component.Git	Git for Windows	16.0.28625.61
Microsoft.VisualStudio.Component.GraphDocument	DGML editor	16.0.28625.61
Microsoft.VisualStudio.Component.LinqToSql	LINQ to SQL tools	16.0.28625.61
Microsoft.VisualStudio.Component.VC.14.20.ARM	MSVC v142 - VS 2019 C++ ARM build tools (v14.20)	16.4.29511.114
Microsoft.VisualStudio.Component.VC.14.20.ARM.Spectre	MSVC v142 - VS 2019 C++ ARM Spectre-mitigated libs (v14.20)	16.4.29511.114
Microsoft.VisualStudio.Component.VC.14.20.ARM64	MSVC v142 - VS 2019 C++ ARM64 build tools (v14.20)	16.4.29511.114
Microsoft.VisualStudio.Component.VC.14.20.ARM64.Spectre	MSVC v142 - VS 2019 C++ ARM64 Spectre-mitigated libs (v14.20)	16.4.29511.114
Microsoft.VisualStudio.Component.VC.14.20.ATL	C++ v14.20 ATL for v142 build tools (x86 & x64)	16.1.28829.92

Component ID	Name	Version
Microsoft.VisualStudio.Component.VC.14.20.ATL.ARM	C++ v14.20 ATL for v142 build tools (ARM)	16.4.29511.114
Microsoft.VisualStudio.Component.VC.14.20.ATL.ARM.Spectre	C++ v14.20 ATL for v142 build tools with Spectre Mitigations (ARM)	16.4.29511.114
Microsoft.VisualStudio.Component.VC.14.20.ATL.ARM64	C++ v14.20 ATL for v142 build tools (ARM64)	16.4.29511.114
Microsoft.VisualStudio.Component.VC.14.20.ATL.ARM64.Spectre	C++ v14.20 ATL for v142 build tools with Spectre Mitigations (ARM64)	16.4.29511.114
Microsoft.VisualStudio.Component.VC.14.20.ATL.Spectre	C++ v14.20 ATL for v142 build tools with Spectre Mitigations (x86 & x64)	16.4.29511.114
Microsoft.VisualStudio.Component.VC.14.20.CLI.Support	C++/CLI support for v142 build tools (14.20)	16.4.29409.204
Microsoft.VisualStudio.Component.VC.14.20.MFC	C++ v14.20 MFC for v142 build tools (x86 & x64)	16.2.29003.222
Microsoft.VisualStudio.Component.VC.14.20.MFC.ARM	C++ v14.20 MFC for v142 build tools (ARM)	16.4.29511.114
Microsoft.VisualStudio.Component.VC.14.20.MFC.ARM.Spectre	C++ v14.20 MFC for v142 build tools with Spectre Mitigations (ARM)	16.4.29511.114
Microsoft.VisualStudio.Component.VC.14.20.MFC.ARM64	C++ v14.20 MFC for v142 build tools (ARM64)	16.4.29511.114
Microsoft.VisualStudio.Component.VC.14.20.MFC.ARM64.Spectre	C++ v14.20 MFC for v142 build tools with Spectre Mitigations (ARM64)	16.4.29511.114
Microsoft.VisualStudio.Component.VC.14.20.MFC.Spectre	C++ v14.20 MFC for v142 build tools with Spectre Mitigations (x86 & x64)	16.4.29511.114
Microsoft.VisualStudio.Component.VC.14.20.x86.x64	MSVC v142 - VS 2019 C++ x64/x86 build tools (v14.20)	16.4.29511.114
Microsoft.VisualStudio.Component.VC.14.20.x86.x64.Spectre	MSVC v142 - VS 2019 C++ x64/x86 Spectre-mitigated libs (v14.20)	16.4.29511.114
Microsoft.VisualStudio.Component.VC.14.21.ARM	MSVC v142 - VS 2019 C++ ARM build tools (v14.21)	16.3.29207.166
Microsoft.VisualStudio.Component.VC.14.21.ARM.Spectre	MSVC v142 - VS 2019 C++ ARM Spectre-mitigated libs (v14.21)	16.4.29429.68
Microsoft.VisualStudio.Component.VC.14.21.ARM64	MSVC v142 - VS 2019 C++ ARM64 build tools (v14.21)	16.3.29207.166
Microsoft.VisualStudio.Component.VC.14.21.ARM64.Spectre	MSVC v142 - VS 2019 C++ ARM64 Spectre-mitigated libs (v14.21)	16.4.29429.68

Component ID	Name	Version
Microsoft.VisualStudio.Component.VC.14.21.ATL	C++ v14.21 ATL for v142 build tools (x86 & x64)	16.2.29019.55
Microsoft.VisualStudio.Component.VC.14.21.ATL.ARM	C++ v14.21 ATL for v142 build tools (ARM)	16.2.29019.55
Microsoft.VisualStudio.Component.VC.14.21.ATL.ARM.Spectre	C++ v14.21 ATL for v142 build tools with Spectre Mitigations (ARM)	16.4.29429.68
Microsoft.VisualStudio.Component.VC.14.21.ATL.ARM64	C++ v14.21 ATL for v142 build tools (ARM64)	16.2.29019.55
Microsoft.VisualStudio.Component.VC.14.21.ATL.ARM64.Spectre	C++ v14.21 ATL for v142 build tools with Spectre Mitigations (ARM64)	16.4.29429.68
Microsoft.VisualStudio.Component.VC.14.21.ATL.Spectre	C++ v14.21 ATL for v142 build tools with Spectre Mitigations (x86 & x64)	16.4.29429.68
Microsoft.VisualStudio.Component.VC.14.21.CLI.Support	C++/CLI support for v142 build tools (14.21)	16.3.29207.166
Microsoft.VisualStudio.Component.VC.14.21.MFC	C++ v14.21 MFC for v142 build tools (x86 & x64)	16.2.29019.55
Microsoft.VisualStudio.Component.VC.14.21.MFC.ARM	C++ v14.21 MFC for v142 build tools (ARM)	16.2.29019.55
Microsoft.VisualStudio.Component.VC.14.21.MFC.ARM.Spectre	C++ v14.21 MFC for v142 build tools with Spectre Mitigations (ARM)	16.4.29429.68
Microsoft.VisualStudio.Component.VC.14.21.MFC.ARM64	C++ v14.21 MFC for v142 build tools (ARM64)	16.2.29019.55
Microsoft.VisualStudio.Component.VC.14.21.MFC.ARM64.Spectre	C++ v14.21 MFC for v142 build tools with Spectre Mitigations (ARM64)	16.4.29429.68
Microsoft.VisualStudio.Component.VC.14.21.MFC.Spectre	C++ v14.21 MFC for v142 build tools with Spectre Mitigations (x86 & x64)	16.4.29429.68
Microsoft.VisualStudio.Component.VC.14.21.x86.x64	MSVC v142 - VS 2019 C++ x64/x86 build tools (v14.21)	16.3.29207.166
Microsoft.VisualStudio.Component.VC.14.21.x86.x64.Spectre	MSVC v142 - VS 2019 C++ x64/x86 Spectre-mitigated libs (v14.21)	16.4.29429.68
Microsoft.VisualStudio.Component.VC.14.22.ARM	MSVC v142 - VS 2019 C++ ARM build tools (v14.22)	16.4.29313.120
Microsoft.VisualStudio.Component.VC.14.22.ARM.Spectre	MSVC v142 - VS 2019 C++ ARM Spectre-mitigated libs (v14.22)	16.4.29429.68
Microsoft.VisualStudio.Component.VC.14.22.ARM64	MSVC v142 - VS 2019 C++ ARM64 build tools (v14.22)	16.4.29313.120

Component ID	Name	Version
Microsoft.VisualStudio.Component.VC.14.22.ARM64.Spectre	MSVC v142 - VS 2019 C++ ARM64 Spectre-mitigated libs (v14.22)	16.4.29429.68
Microsoft.VisualStudio.Component.VC.14.22.ATL	C++ v14.22 ATL for v142 build tools (x86 & x64)	16.4.29313.120
Microsoft.VisualStudio.Component.VC.14.22.ATL.ARM	C++ v14.22 ATL for v142 build tools (ARM)	16.4.29313.120
Microsoft.VisualStudio.Component.VC.14.22.ATL.ARM.Spectre	C++ v14.22 ATL for v142 build tools with Spectre Mitigations (ARM)	16.4.29429.68
Microsoft.VisualStudio.Component.VC.14.22.ATL.ARM64	C++ v14.22 ATL for v142 build tools (ARM64)	16.4.29313.120
Microsoft.VisualStudio.Component.VC.14.22.ATL.ARM64.Spectre	C++ v14.22 ATL for v142 build tools with Spectre Mitigations (ARM64)	16.4.29429.68
Microsoft.VisualStudio.Component.VC.14.22.ATL.Spectre	C++ v14.22 ATL for v142 build tools with Spectre Mitigations (x86 & x64)	16.4.29429.68
Microsoft.VisualStudio.Component.VC.14.22.CLI.Support	C++/CLI support for v142 build tools (14.22)	16.4.29313.120
Microsoft.VisualStudio.Component.VC.14.22.MFC	C++ v14.22 MFC for v142 build tools (x86 & x64)	16.4.29313.120
Microsoft.VisualStudio.Component.VC.14.22.MFC.ARM	C++ v14.22 MFC for v142 build tools (ARM)	16.4.29313.120
Microsoft.VisualStudio.Component.VC.14.22.MFC.ARM.Spectre	C++ v14.22 MFC for v142 build tools with Spectre Mitigations (ARM)	16.4.29429.68
Microsoft.VisualStudio.Component.VC.14.22.MFC.ARM64	C++ v14.22 MFC for v142 build tools (ARM64)	16.4.29313.120
Microsoft.VisualStudio.Component.VC.14.22.MFC.ARM64.Spectre	C++ v14.22 MFC for v142 build tools with Spectre Mitigations (ARM64)	16.4.29429.68
Microsoft.VisualStudio.Component.VC.14.22.MFC.Spectre	C++ v14.22 MFC for v142 build tools with Spectre Mitigations (x86 & x64)	16.4.29429.68
Microsoft.VisualStudio.Component.VC.14.22.x86.x64	MSVC v142 - VS 2019 C++ x64/x86 build tools (v14.22)	16.4.29313.120
Microsoft.VisualStudio.Component.VC.14.22.x86.x64.Spectre	MSVC v142 - VS 2019 C++ x64/x86 Spectre-mitigated libs (v14.22)	16.4.29429.68
Microsoft.VisualStudio.Component.VC.14.23.ARM	MSVC v142 - VS 2019 C++ ARM build tools (v14.23)	16.4.29429.68
Microsoft.VisualStudio.Component.VC.14.23.ARM.Spectre	MSVC v142 - VS 2019 C++ ARM Spectre-mitigated libs (v14.23)	16.4.29429.68

Component ID	Name	Version
Microsoft.VisualStudio.Component.VC.14.23.ARM64	MSVC v142 - VS 2019 C++ ARM64 build tools (v14.23)	16.4.29429.68
Microsoft.VisualStudio.Component.VC.14.23.ARM64.Spectre	MSVC v142 - VS 2019 C++ ARM64 Spectre-mitigated libs (v14.23)	16.4.29429.68
Microsoft.VisualStudio.Component.VC.14.23.ATL	C++ v14.23 ATL for v142 build tools (x86 & x64)	16.4.29429.68
Microsoft.VisualStudio.Component.VC.14.23.ATL.ARM	C++ v14.23 ATL for v142 build tools (ARM)	16.4.29429.68
Microsoft.VisualStudio.Component.VC.14.23.ATL.ARM.Spectre	C++ v14.23 ATL for v142 build tools with Spectre Mitigations (ARM)	16.4.29429.68
Microsoft.VisualStudio.Component.VC.14.23.ATL.ARM64	C++ v14.23 ATL for v142 build tools (ARM64)	16.4.29429.68
Microsoft.VisualStudio.Component.VC.14.23.ATL.ARM64.Spectre	C++ v14.23 ATL for v142 build tools with Spectre Mitigations (ARM64)	16.4.29429.68
Microsoft.VisualStudio.Component.VC.14.23.ATL.Spectre	C++ v14.23 ATL for v142 build tools with Spectre Mitigations (x86 & x64)	16.4.29429.68
Microsoft.VisualStudio.Component.VC.14.23.CLI.Support	C++/CLI support for v142 build tools (14.23)	16.4.29409.204
Microsoft.VisualStudio.Component.VC.14.23.MFC	C++ v14.23 MFC for v142 build tools (x86 & x64)	16.4.29429.68
Microsoft.VisualStudio.Component.VC.14.23.MFC.ARM	C++ v14.23 MFC for v142 build tools (ARM)	16.4.29429.68
Microsoft.VisualStudio.Component.VC.14.23.MFC.ARM.Spectre	C++ v14.23 MFC for v142 build tools with Spectre Mitigations (ARM)	16.4.29429.68
Microsoft.VisualStudio.Component.VC.14.23.MFC.ARM64	C++ v14.23 MFC for v142 build tools (ARM64)	16.4.29429.68
Microsoft.VisualStudio.Component.VC.14.23.MFC.ARM64.Spectre	C++ v14.23 MFC for v142 build tools with Spectre Mitigations (ARM64)	16.4.29429.68
Microsoft.VisualStudio.Component.VC.14.23.MFC.Spectre	C++ v14.23 MFC for v142 build tools with Spectre Mitigations (x86 & x64)	16.4.29429.68
Microsoft.VisualStudio.Component.VC.14.23.x86.x64	MSVC v142 - VS 2019 C++ x64/x86 build tools (v14.23)	16.4.29429.68
Microsoft.VisualStudio.Component.VC.14.23.x86.x64.Spectre	MSVC v142 - VS 2019 C++ x64/x86 Spectre-mitigated libs (v14.23)	16.4.29429.68
Microsoft.VisualStudio.Component.VC.ATL.ARM	C++ ATL for latest v142 build tools (ARM)	16.4.29313.120

Component ID	Name	Version
Microsoft.VisualStudio.Component.VC.ATL.ARM.Spectre	C++ ATL for latest v142 build tools with Spectre Mitigations (ARM)	16.4.29429.68
Microsoft.VisualStudio.Component.VC.ATL.ARM64	C++ ATL for latest v142 build tools (ARM64)	16.4.29313.120
Microsoft.VisualStudio.Component.VC.ATL.ARM64.Spectre	C++ ATL for latest v142 build tools with Spectre Mitigations (ARM64)	16.4.29429.68
Microsoft.VisualStudio.Component.VC.ATL.Spectre	C++ ATL for latest v142 build tools with Spectre Mitigations (x86 & x64)	16.4.29429.68
Microsoft.VisualStudio.Component.VC.ATLMFC.Spectre	C++ MFC for latest v142 build tools with Spectre Mitigations (x86 & x64)	16.4.29429.68
Microsoft.VisualStudio.Component.VC.MFC.ARM	C++ MFC for latest v142 build tools (ARM)	16.4.29313.120
Microsoft.VisualStudio.Component.VC.MFC.ARM.Spectre	C++ MFC for latest v142 build tools with Spectre Mitigations (ARM)	16.4.29429.68
Microsoft.VisualStudio.Component.VC.MFC.ARM64	C++ MFC for latest v142 build tools (ARM64)	16.4.29313.120
Microsoft.VisualStudio.Component.VC.MFC.ARM64.Spectre	C++ MFC for latest v142 build tools with Spectre Mitigations (ARM64)	16.4.29429.68
Microsoft.VisualStudio.Component.VC.Redist.MSM	C++ 2019 Redistributable MSMs	16.4.29429.68
Microsoft.VisualStudio.Component.VC.Runtimes.ARM.Spectre	MSVC v142 - VS 2019 C++ ARM Spectre-mitigated libs (v14.24)	16.4.29429.68
Microsoft.VisualStudio.Component.VC.Runtimes.ARM64.Spectre	MSVC v142 - VS 2019 C++ ARM64 Spectre-mitigated libs (v14.24)	16.4.29429.68
Microsoft.VisualStudio.Component.VC.Runtimes.x86.x64.Spectre	MSVC v142 - VS 2019 C++ x64/x86 Spectre-mitigated libs (v14.24)	16.4.29429.68
Microsoft.VisualStudio.Component.VC.v141.ARM.Spectre	MSVC v141 - VS 2017 C++ ARM Spectre-mitigated libs (v14.16)	16.4.29429.68
Microsoft.VisualStudio.Component.VC.v141.ARM64.Spectre	MSVC v141 - VS 2017 C++ ARM64 Spectre-mitigated libs (v14.16)	16.4.29429.68
Microsoft.VisualStudio.Component.VC.v141.ATL	C++ ATL for v141 build tools (x86 & x64)	16.0.28625.61
Microsoft.VisualStudio.Component.VC.v141.ATL.ARM	C++ ATL for v141 build tools (ARM)	16.0.28625.61
Microsoft.VisualStudio.Component.VC.v141.ATL.ARM.Spectre	C++ ATL for v141 build tools with Spectre Mitigations (ARM)	16.0.28625.61

Component ID	Name	Version
Microsoft.VisualStudio.Component.VC.v141.ATL.ARM64	C++ ATL for v141 build tools (ARM64)	16.0.28625.61
Microsoft.VisualStudio.Component.VC.v141.ATL.ARM64.Spectre	C++ ATL for v141 build tools with Spectre Mitigations (ARM64)	16.0.28625.61
Microsoft.VisualStudio.Component.VC.v141.ATL.Spectre	C++ ATL for v141 build tools with Spectre Mitigations (x86 & x64)	16.0.28625.61
Microsoft.VisualStudio.Component.VC.v141.CLI.Support	C++/CLI support for v141 build tools (14.16)	16.3.29207.166
Microsoft.VisualStudio.Component.VC.v141.MFC	C++ MFC for v141 build tools (x86 & x64)	16.0.28625.61
Microsoft.VisualStudio.Component.VC.v141.MFC.ARM	C++ MFC for v141 build tools (ARM)	16.2.28915.88
Microsoft.VisualStudio.Component.VC.v141.MFC.ARM.Spectre	C++ MFC for v141 build tools with Spectre Mitigations (ARM)	16.0.28625.61
Microsoft.VisualStudio.Component.VC.v141.MFC.ARM64	C++ MFC for v141 build tools (ARM64)	16.0.28625.61
Microsoft.VisualStudio.Component.VC.v141.MFC.ARM64.Spectre	C++ MFC for v141 build tools with Spectre Mitigations (ARM64)	16.0.28625.61
Microsoft.VisualStudio.Component.VC.v141.MFC.Spectre	C++ MFC for v141 build tools with Spectre Mitigations (x86 & x64)	16.0.28625.61
Microsoft.VisualStudio.Component.VC.v141.x86.x64.Spectre	MSVC v141 - VS 2017 C++ x64/x86 Spectre-mitigated libs (v14.16)	16.4.29429.68
Microsoft.VisualStudio.Component.VisualStudioData	Data sources and service references	16.0.28707.177
Microsoft.VisualStudio.Component.WinXP	C++ Windows XP Support for VS 2017 (v141) tools [Deprecated]	16.1.28811.260
Microsoft.VisualStudio.Web.Mvc4.ComponentGroup	ASP.NET MVC 4	16.1.28810.153

Get support

Sometimes, things can go wrong. If your Visual Studio installation fails, see [Troubleshoot Visual Studio installation and upgrade issues](#) for step-by-step guidance.

We also offer a [live chat](#) (English only) support option for installation-related issues.

Here are a few more support options:

- Report product issues to us via the [Report a Problem](#) tool that appears both in the Visual Studio Installer and in the Visual Studio IDE.
- Suggest a feature, track product issues, and find answers in the [Visual Studio Developer Community](#).

- Use your [GitHub](#) account to talk to us and other Visual Studio developers in the [Visual Studio conversation in the Gitter community](#).

See also

- [Visual Studio workload and component IDs](#)
- [Visual Studio administrator guide](#)
- [Use command-line parameters to install Visual Studio](#)
 - [Command-line parameter examples](#)
- [Create an offline installation of Visual Studio](#)

Visual Studio Community component directory

12/4/2019 • 67 minutes to read • [Edit Online](#)

The tables on this page list the IDs that you can use to install Visual Studio from the command line, or that you can specify as a dependency in a VSIX manifest. We'll add additional components when we release updates to Visual Studio.

Also note the following about the page:

- Each workload has its own section, followed by the workload ID and a table of the components that are available for the workload.
- By default, the **Required** components will be installed when you install the workload.
- If you choose to, you can also install the **Recommended** and **Optional** components.
- We've also added a section that lists the additional components that are not affiliated with any workload.

When you set dependencies in your VSIX manifest, you must specify Component IDs only. Use the tables on this page to determine the minimum component dependencies. In some scenarios, this might mean that you specify only one component from a workload. In other scenarios, it might mean that you specify multiple components from a single workload or multiple components from multiple workloads. For more information, see the [How to: Migrate extensibility projects to Visual Studio 2017](#) page.

For more information about how to use these IDs, see the [Use command-line parameters to install Visual Studio 2017](#) page. And, for a list of workload and component IDs for other products, see the [Visual Studio workload and component IDs](#) page.

Visual Studio core editor (included with Visual Studio Community 2017)

ID: Microsoft.VisualStudio.Workload.CoreEditor

Description: The Visual Studio core shell experience, including syntax-aware code editing, source code control and work item management.

Components included by this workload

COMPONENT ID	NAME	VERSION	DEPENDENCY TYPE
Microsoft.VisualStudio.Component.CoreEditor	Visual Studio core editor	15.8.27729.1	Required
Microsoft.VisualStudio.Component.StartPageExperiment.Cpp	Visual Studio Start Page for C++ Users	15.0.27128.1	Optional

Azure development

ID: Microsoft.VisualStudio.Workload.Azure

Description: Azure SDKs, tools, and projects for developing cloud apps, creating resources, and building Containers including Docker support.

Components included by this workload

Component ID	Name	Version	Dependency Type
Component.Microsoft.VisualStudio.RazorExtension	Razor Language Services	15.0.26720.2	Required
Component.Microsoft.VisualStudio.Web.AzureFunctions	Microsoft Azure WebJobs Tools	15.7.27617.1	Required
Component.Microsoft.Web.LibraryManager	Library Manager	15.8.27705.0	Required
Component.WebSocket	WebSocket4Net	15.0.26606.0	Required
Microsoft.Component.ClickOnce	ClickOnce Publishing	15.8.27825.0	Required
Microsoft.Component.MSBuild	MSBuild	15.7.27520.0	Required
Microsoft.Component.NetFX.Core.Runtime	.NET Core runtime	15.0.26208.0	Required
Microsoft.Net.Component.4.5.2.TargetingPack	.NET Framework 4.5.2 targeting pack	15.6.27406.0	Required
Microsoft.Net.Component.4.5.TargetingPack	.NET Framework 4.5 targeting pack	15.6.27406.0	Required
Microsoft.Net.Component.4.6.1.SDK	.NET Framework 4.6.1 SDK	15.6.27406.0	Required
Microsoft.Net.Component.4.6.1.TargetingPack	.NET Framework 4.6.1 targeting pack	15.6.27406.0	Required
Microsoft.Net.ComponentGroup.DevelopmentPrerequisites	.NET Framework 4.6.1 development tools	15.8.27825.0	Required
Microsoft.Net.Core.Component.SDK.2.1	.NET Core 2.1 development tools	15.8.27924.0	Required
Microsoft.NetCore.ComponentGroup.DevelopmentTools.2.1	.NET Core 2.1 development tools	15.8.27924.0	Required
Microsoft.NetCore.ComponentGroup.Web.2.1	.NET Core 2.1 development tools	15.8.27924.0	Required
Microsoft.VisualStudio.Component.Azure.AuthoringTools	Azure Authoring Tools	15.9.28307.421	Required
Microsoft.VisualStudio.Component.Azure.ClientLibs	Azure libraries for .NET	15.0.26208.0	Required

Component ID	Name	Version	Dependency Type
Microsoft.VisualStudio.Component.Azure.Compute Emulator	Azure Compute Emulator	15.9.28307.421	Required
Microsoft.VisualStudio.Component.Azure.Storage.Emulator	Azure Storage Emulator	15.9.28125.51	Required
Microsoft.VisualStudio.Component.CloudExplorer	Cloud Explorer	15.9.28230.55	Required
Microsoft.VisualStudio.Component.Common.Azure.Tools	Connectivity and publishing tools	15.9.28107.0	Required
Microsoft.VisualStudio.Component.DockerTools	Container development tools	15.8.27906.1	Required
Microsoft.VisualStudio.Component.DockerTools.BuildTools	Container development tools - Build Tools	15.7.27617.1	Required
Microsoft.VisualStudio.Component.FSharp	F# language support	15.8.27825.0	Required
Microsoft.VisualStudio.Component.FSharp.WebTemplates	F# language support for web projects	15.9.28307.421	Required
Microsoft.VisualStudio.Component.IISExpress	IIS Express	15.0.26208.0	Required
Microsoft.VisualStudio.Component.JavaScript.Diagnostics	JavaScript diagnostics	15.8.27729.1	Required
Microsoft.VisualStudio.Component.JavaScript.TypeScript	JavaScript and TypeScript language support	15.9.28125.51	Required
Microsoft.VisualStudio.Component.ManagedDesktop.Core	Managed Desktop Workload Core	15.8.27729.1	Required
Microsoft.VisualStudio.Component.NuGet	NuGet package manager	15.9.28016.0	Required
Microsoft.VisualStudio.Component.PortableLibrary	.NET Portable Library targeting pack	15.6.27309.0	Required
Microsoft.VisualStudio.Component.Roslyn.Compiler	C# and Visual Basic Roslyn compilers	15.6.27309.0	Required
Microsoft.VisualStudio.Component.Roslyn.LanguageServices	C# and Visual Basic	15.8.27729.1	Required

Component ID	Name	Version	Dependency Type
Microsoft.VisualStudio.Component.SQL.ADAL	SQL ADAL runtime	15.6.27406.0	Required
Microsoft.VisualStudio.Component.SQL.CLR	CLR data types for SQL Server	15.0.26208.0	Required
Microsoft.VisualStudio.Component.SQL.CMDUtils	SQL Server Command Line Utilities	15.0.26208.0	Required
Microsoft.VisualStudio.Component.SQL.DataSources	Data sources for SQL Server support	15.0.26621.2	Required
Microsoft.VisualStudio.Component.SQL.LocalDB.Runtime	SQL Server Express 2016 LocalDB	15.7.27617.1	Required
Microsoft.VisualStudio.Component.SQL.NCLI	SQL Server Native Client	15.0.26208.0	Required
Microsoft.VisualStudio.Component.SQL.SSDT	SQL Server Data Tools	15.9.28107.0	Required
Microsoft.VisualStudio.Component.Static.Analysis.Tools	Static analysis tools	15.0.26208.0	Required
Microsoft.VisualStudio.Component.TextTemplating	Text Template Transformation	15.0.26208.0	Required
Microsoft.VisualStudio.Component.TypeScript.3.1	TypeScript 3.1 SDK	15.0.28218.60	Required
Microsoft.VisualStudio.Component.VisualStudioData	Data sources and service references	15.6.27406.0	Required
Microsoft.VisualStudio.Component.Web	ASP.NET and web development tools	15.8.27825.0	Required
Microsoft.VisualStudio.ComponentGroup.Azure.Prerequisites	Azure development prerequisites	15.9.28107.0	Required
Microsoft.VisualStudio.ComponentGroup.AzureFunctions	Microsoft Azure WebJobs Tools	15.7.27617.1	Required
Microsoft.VisualStudio.ComponentGroup.Web	ASP.NET and web development tools prerequisites	15.9.28219.51	Required
Microsoft.VisualStudio.ComponentGroup.WebToolsExtensions	ASP.NET and web development	15.8.27825.0	Required
Microsoft.Component.Azure.DataLake.Tools	Azure Data Lake and Stream Analytics Tools	15.9.28107.0	Recommended

Component ID	Name	Version	Dependency Type
Microsoft.Net.Component.4.5.1.TargetingPack	.NET Framework 4.5.1 targeting pack	15.6.27406.0	Recommended
Microsoft.Net.Component.4.6.TargetingPack	.NET Framework 4.6 targeting pack	15.6.27406.0	Recommended
Microsoft.Net.Component.4.TargetingPack	.NET Framework 4 targeting pack	15.6.27406.0	Recommended
Microsoft.ComponentGroup.TargetingPacks.Common	.NET Framework 4 – 4.6 development tools	15.6.27406.0	Recommended
Microsoft.VisualStudio.Component.AspNet45	Advanced ASP.NET features	15.7.27625.0	Recommended
Microsoft.VisualStudio.Component.Azure.MobileAppsSdk	Azure Mobile Apps SDK	15.7.27625.0	Recommended
Microsoft.VisualStudio.Component.Azure.ResourceManager.Tools	Azure Resource Manager core tools	15.9.28107.0	Recommended
Microsoft.VisualStudio.Component.Azure.ServiceFabric.Tools	Service Fabric Tools	15.8.27825.0	Recommended
Microsoft.VisualStudio.Component.Azure.Waverton	Azure Cloud Services core tools	15.9.28107.0	Recommended
Microsoft.VisualStudio.Component.Azure.Waverton.BuildTools	Azure Cloud Services build tools	15.7.27617.1	Recommended
Microsoft.VisualStudio.Component.DiagnosticTools	.NET profiling tools	15.8.27729.1	Recommended
Microsoft.VisualStudio.Component.WebDeploy	Web Deploy	15.8.27729.1	Recommended
Microsoft.VisualStudio.ComponentGroup.Azure.CloudServices	Azure Cloud Services tools	15.0.26504.0	Recommended
Microsoft.VisualStudio.ComponentGroup.Azure.ResourceManager.Tools	Azure Resource Manager tools	15.0.27005.2	Recommended
Microsoft.Net.Component.4.6.2.SDK	.NET Framework 4.6.2 SDK	15.6.27406.0	Optional
Microsoft.Net.Component.4.6.2.TargetingPack	.NET Framework 4.6.2 targeting pack	15.6.27406.0	Optional

Component ID	Name	Version	Dependency Type
Microsoft.Net.Component.4.7.1.SDK	.NET Framework 4.7.1 SDK	15.6.27406.0	Optional
Microsoft.Net.Component.4.7.1.TargetingPack	.NET Framework 4.7.1 targeting pack	15.6.27406.0	Optional
Microsoft.Net.Component.4.7.2.SDK	.NET Framework 4.7.2 SDK	15.8.27825.0	Optional
Microsoft.Net.Component.4.7.2.TargetingPack	.NET Framework 4.7.2 targeting pack	15.8.27825.0	Optional
Microsoft.Net.Component.4.7.SDK	.NET Framework 4.7 SDK	15.6.27406.0	Optional
Microsoft.Net.Component.4.7.TargetingPack	.NET Framework 4.7 targeting pack	15.6.27406.0	Optional
Microsoft.Net.ComponentGroup.4.6.2.DeveloperTools	.NET Framework 4.6.2 development tools	15.6.27406.0	Optional
Microsoft.Net.ComponentGroup.4.7.1.DeveloperTools	.NET Framework 4.7.1 development tools	15.6.27406.0	Optional
Microsoft.Net.ComponentGroup.4.7.2.DeveloperTools	.NET Framework 4.7.2 development tools	15.8.27825.0	Optional
Microsoft.Net.ComponentGroup.4.7.DeveloperTools	.NET Framework 4.7 development tools	15.6.27406.0	Optional
Microsoft.Net.Core.Component.SDK	.NET Core 2.0 development tools	15.6.27406.0	Optional
Microsoft.Net.Core.Component.SDK.1x	.NET Core 1.0 - 1.1 development tools	15.6.27406.0	Optional
Microsoft.NetCore.1x.ComponentGroup.Web	.NET Core 1.0 - 1.1 development tools for Web	15.6.27406.0	Optional
Microsoft.NetCore.ComponentGroup.DevelopmentTools	.NET Core 2.0 development tools	15.8.27729.1	Optional
Microsoft.NetCore.ComponentGroup.Web	.NET Core 2.0 development tools	15.7.27625.0	Optional
Microsoft.VisualStudio.Component.Azure.Storage.AzCopy	Azure Storage AzCopy	15.0.26906.1	Optional
Microsoft.VisualStudio.Component.Wcf.Tooling	Windows Communication Foundation	15.8.27924.0	Optional

Data storage and processing

ID: Microsoft.VisualStudio.Workload.Data

Description: Connect, develop, and test data solutions with SQL Server, Azure Data Lake, or Hadoop.

Components included by this workload

Component ID	Name	Version	Dependency Type
Component.Microsoft.VisualStudio.RazorExtension	Razor Language Services	15.0.26720.2	Recommended
Component.Microsoft.Web.LibraryManager	Library Manager	15.8.27705.0	Recommended
Component.Redgate.SQLSearch.VSExtension	Redgate SQL Search	3.1.7.2062	Recommended
Component.WebSocket	WebSocket4Net	15.0.26606.0	Recommended
Microsoft.Component.Azure.DataLake.Tools	Azure Data Lake and Stream Analytics Tools	15.9.28107.0	Recommended
Microsoft.Component.ClickOnce	ClickOnce Publishing	15.8.27825.0	Recommended
Microsoft.Component.MSBuild	MSBuild	15.7.27520.0	Recommended
Microsoft.Net.Component.4.5.1.TargetingPack	.NET Framework 4.5.1 targeting pack	15.6.27406.0	Recommended
Microsoft.Net.Component.4.5.2.TargetingPack	.NET Framework 4.5.2 targeting pack	15.6.27406.0	Recommended
Microsoft.Net.Component.4.5.TargetingPack	.NET Framework 4.5 targeting pack	15.6.27406.0	Recommended
Microsoft.Net.Component.4.6.1.SDK	.NET Framework 4.6.1 SDK	15.6.27406.0	Recommended
Microsoft.Net.Component.4.6.1.TargetingPack	.NET Framework 4.6.1 targeting pack	15.6.27406.0	Recommended
Microsoft.Net.Component.4.6.TargetingPack	.NET Framework 4.6 targeting pack	15.6.27406.0	Recommended
Microsoft.Net.Component.4.TargetingPack	.NET Framework 4 targeting pack	15.6.27406.0	Recommended
Microsoft.Net.ComponentGroup.DevelopmentPrerequisites	.NET Framework 4.6.1 development tools	15.8.27825.0	Recommended
Microsoft.Net.ComponentGroup.TargetingPacks.Common	.NET Framework 4 – 4.6 development tools	15.6.27406.0	Recommended

Component ID	Name	Version	Dependency Type
Microsoft.Net.Core.Component.SDK.2.1	.NET Core 2.1 development tools	15.8.27924.0	Recommended
Microsoft.VisualStudio.Component.Azure.AuthoringTools	Azure Authoring Tools	15.9.28307.421	Recommended
Microsoft.VisualStudio.Component.Azure.ClientLibs	Azure libraries for .NET	15.0.26208.0	Recommended
Microsoft.VisualStudio.Component.Azure.Compute.Emulator	Azure Compute Emulator	15.9.28307.421	Recommended
Microsoft.VisualStudio.Component.Azure.Storage.Emulator	Azure Storage Emulator	15.9.28125.51	Recommended
Microsoft.VisualStudio.Component.Azure.Waverton	Azure Cloud Services core tools	15.9.28107.0	Recommended
Microsoft.VisualStudio.Component.Azure.Waverton.BuildTools	Azure Cloud Services build tools	15.7.27617.1	Recommended
Microsoft.VisualStudio.Component.CloudExplorer	Cloud Explorer	15.9.28230.55	Recommended
Microsoft.VisualStudio.Component.Common.Azure.Tools	Connectivity and publishing tools	15.9.28107.0	Recommended
Microsoft.VisualStudio.Component.DockerTools	Container development tools	15.8.27906.1	Recommended
Microsoft.VisualStudio.Component.DockerTools.BuildTools	Container development tools - Build Tools	15.7.27617.1	Recommended
Microsoft.VisualStudio.Component.IISExpress	IIS Express	15.0.26208.0	Recommended
Microsoft.VisualStudio.Component.JavaScript.Diagnostics	JavaScript diagnostics	15.8.27729.1	Recommended
Microsoft.VisualStudio.Component.JavaScript.TypeScript	JavaScript and TypeScript language support	15.9.28125.51	Recommended
Microsoft.VisualStudio.Component.ManagedDesktop.Core	Managed Desktop Workload Core	15.8.27729.1	Recommended
Microsoft.VisualStudio.Component.NuGet	NuGet package manager	15.9.28016.0	Recommended

Component ID	Name	Version	Dependency Type
Microsoft.VisualStudio.Component.PortableLibrary	.NET Portable Library targeting pack	15.6.27309.0	Recommended
Microsoft.VisualStudio.Component.Roslyn.Compiler	C# and Visual Basic Roslyn compilers	15.6.27309.0	Recommended
Microsoft.VisualStudio.Component.Roslyn.LanguageServices	C# and Visual Basic	15.8.27729.1	Recommended
Microsoft.VisualStudio.Component.SQL.ADAL	SQL ADAL runtime	15.6.27406.0	Recommended
Microsoft.VisualStudio.Component.SQL.CLR	CLR data types for SQL Server	15.0.26208.0	Recommended
Microsoft.VisualStudio.Component.SQL.CMDUtils	SQL Server Command Line Utilities	15.0.26208.0	Recommended
Microsoft.VisualStudio.Component.SQL.DataSources	Data sources for SQL Server support	15.0.26621.2	Recommended
Microsoft.VisualStudio.Component.SQL.LocalDB.Runtime	SQL Server Express 2016 LocalDB	15.7.27617.1	Recommended
Microsoft.VisualStudio.Component.SQL.NCLI	SQL Server Native Client	15.0.26208.0	Recommended
Microsoft.VisualStudio.Component.SQL.SSDT	SQL Server Data Tools	15.9.28107.0	Recommended
Microsoft.VisualStudio.Component.Static.Analysis.Tools	Static analysis tools	15.0.26208.0	Recommended
Microsoft.VisualStudio.Component.TextTemplating	Text Template Transformation	15.0.26208.0	Recommended
Microsoft.VisualStudio.Component.TypeScript.3.1	TypeScript 3.1 SDK	15.0.28218.60	Recommended
Microsoft.VisualStudio.Component.VisualStudioData	Data sources and service references	15.6.27406.0	Recommended
Microsoft.VisualStudio.Component.Web	ASP.NET and web development tools	15.8.27825.0	Recommended
Microsoft.VisualStudio.ComponentGroup.Web	ASP.NET and web development tools prerequisites	15.9.28219.51	Recommended
Microsoft.VisualStudio.ComponentGroup.WebToolsExtensions	ASP.NET and web development	15.8.27825.0	Recommended

Component ID	Name	Version	Dependency Type
Microsoft.VisualStudio.Component.FSharp.Desktop	F# desktop language support	15.8.27825.0	Optional

Data science and analytical applications

ID: Microsoft.VisualStudio.Workload.DataScience

Description: Languages and tooling for creating data science applications, including Python, R and F#.

Components included by this workload

Component ID	Name	Version	Dependency Type
Component.Anconda3.x64	Anaconda3 64-bit (5.2.0)	5.2.0	Recommended
Microsoft.Component.CookiecutterTools	Cookiecutter template support	15.0.26621.2	Recommended
Microsoft.Component.PythonTools	Python language support	15.0.26823.1	Recommended
Microsoft.Component.PythonTools.Web	Python web support	15.9.28107.0	Recommended
Microsoft.Net.Component.4.6.1.TargetingPack	.NET Framework 4.6.1 targeting pack	15.6.27406.0	Recommended
Microsoft.VisualStudio.Component.Common.Azure.Tools	Connectivity and publishing tools	15.9.28107.0	Recommended
Microsoft.VisualStudio.Component.FSharp.Desktop	F# desktop language support	15.8.27825.0	Recommended
Microsoft.VisualStudio.Component.JavaScript.TypeScript	JavaScript and TypeScript language support	15.9.28125.51	Recommended
Microsoft.VisualStudio.Component.NuGet	NuGet package manager	15.9.28016.0	Recommended
Microsoft.VisualStudio.Component.R.Open	Microsoft R Client (3.3.2)	15.6.27406.0	Recommended
Microsoft.VisualStudio.Component.RHost	Runtime support for R development tools	15.6.27406.0	Recommended
Microsoft.VisualStudio.Component.Roslyn.Compiler	C# and Visual Basic Roslyn compilers	15.6.27309.0	Recommended
Microsoft.VisualStudio.Component.Roslyn.LanguageServices	C# and Visual Basic	15.8.27729.1	Recommended
Microsoft.VisualStudio.Component.RTools	R language support	15.0.26919.1	Recommended

Component ID	Name	Version	Dependency Type
Microsoft.VisualStudio.Component.SQL CLR	CLR data types for SQL Server	15.0.26208.0	Recommended
Microsoft.VisualStudio.Component.Static.Analysis.Tools	Static analysis tools	15.0.26208.0	Recommended
Microsoft.VisualStudio.Component.TypeScript.3.1	TypeScript 3.1 SDK	15.0.28218.60	Recommended
Microsoft.VisualStudio.Component.VisualStudioData	Data sources and service references	15.6.27406.0	Recommended
Microsoft.VisualStudio.Component.WebDeploy	Web Deploy	15.8.27729.1	Recommended
Microsoft.VisualStudio.ComponentGroup.WebToolsExtensions	ASP.NET and web development	15.8.27825.0	Recommended
Component.Anaconda2.x64	Anaconda2 64-bit (5.2.0)	5.2.0	Optional
Component.Anaconda2.x86	Anaconda2 32-bit (5.2.0)	5.2.0	Optional
Component.Anaconda3.x86	Anaconda3 32-bit (5.2.0)	5.2.0	Optional
Microsoft.Component.VC.Runtime.UCRTSDK	Windows Universal CRT SDK	15.6.27309.0	Optional
Microsoft.ComponentGroup.PythonTools.NativeDevelopment	Python native development tools	15.9.28307.102	Optional
Microsoft.VisualStudio.Component.Graphics.Tools	Graphics debugger and GPU profiler for DirectX	15.6.27406.0	Optional
Microsoft.VisualStudio.Component.Graphics.Win81	Graphics Tools Windows 8.1 SDK	15.6.27406.0	Optional
Microsoft.VisualStudio.Component.VC.140	VC++ 2015.3 v14.00 (v140) toolset for desktop	15.7.27617.1	Optional
Microsoft.VisualStudio.Component.VC.Corelde	Visual Studio C++ core features	15.6.27406.0	Optional
Microsoft.VisualStudio.Component.VC.DiagnosticTools	C++ profiling tools	15.0.26823.1	Optional
Microsoft.VisualStudio.Component.VC.Tools.x86.x64	VC++ 2017 version 15.9 v14.16 latest v141 tools	15.9.28230.55	Optional
Microsoft.VisualStudio.Component.Windows10SDK	Windows Universal C Runtime	15.6.27406.0	Optional

Component ID	Name	Version	Dependency Type
Microsoft.VisualStudio.Component.Windows10SDK.17763	Windows 10 SDK (10.0.17763.0)	15.9.28307.102	Optional
Microsoft.VisualStudio.Component.Windows81SDK	Windows 8.1 SDK	15.6.27406.0	Optional

.NET desktop development

ID: Microsoft.VisualStudio.Workload.ManagedDesktop

Description: Build WPF, Windows Forms, and console applications using C#, Visual Basic, and F#.

Components included by this workload

Component ID	Name	Version	Dependency Type
Microsoft.Component.ClickOnce	ClickOnce Publishing	15.8.27825.0	Required
Microsoft.Component.MSBuild	MSBuild	15.7.27520.0	Required
Microsoft.Net.Component.4.6.1.SDK	.NET Framework 4.6.1 SDK	15.6.27406.0	Required
Microsoft.Net.Component.4.6.1.TargetingPack	.NET Framework 4.6.1 targeting pack	15.6.27406.0	Required
Microsoft.Net.ComponentGroup.DevelopmentPrerequisites	.NET Framework 4.6.1 development tools	15.8.27825.0	Required
Microsoft.VisualStudio.Component.ManagedDesktop.Core	Managed Desktop Workload Core	15.8.27729.1	Required
Microsoft.VisualStudio.Component.ManagedDesktop.Prerequisites	.NET desktop development tools	15.7.27625.0	Required
Microsoft.VisualStudio.Component.PortableLibrary	.NET Portable Library targeting pack	15.6.27309.0	Required
Microsoft.VisualStudio.Component.Roslyn.Compiler	C# and Visual Basic Roslyn compilers	15.6.27309.0	Required
Microsoft.VisualStudio.Component.Roslyn.LanguageServices	C# and Visual Basic	15.8.27729.1	Required
Microsoft.VisualStudio.Component.SQL.CLR	CLR data types for SQL Server	15.0.26208.0	Required

Component ID	Name	Version	Dependency Type
Microsoft.VisualStudio.Component.Static.Analysis.Tools	Static analysis tools	15.0.26208.0	Required
Microsoft.VisualStudio.Component.TextTemplating	Text Template Transformation	15.0.26208.0	Required
Microsoft.VisualStudio.Component.VisualStudioData	Data sources and service references	15.6.27406.0	Required
Microsoft.ComponentGroup.Blend	Blend for Visual Studio	15.6.27406.0	Recommended
Microsoft.Net.Component.4.5.1.TargetingPack	.NET Framework 4.5.1 targeting pack	15.6.27406.0	Recommended
Microsoft.Net.Component.4.5.2.TargetingPack	.NET Framework 4.5.2 targeting pack	15.6.27406.0	Recommended
Microsoft.Net.Component.4.5.TargetingPack	.NET Framework 4.5 targeting pack	15.6.27406.0	Recommended
Microsoft.Net.Component.4.6.TargetingPack	.NET Framework 4.6 targeting pack	15.6.27406.0	Recommended
Microsoft.Net.Component.4.TargetingPack	.NET Framework 4 targeting pack	15.6.27406.0	Recommended
Microsoft.Net.ComponentGroup.TargetingPacks.Common	.NET Framework 4 – 4.6 development tools	15.6.27406.0	Recommended
Microsoft.VisualStudio.Component.Debugger.JustInTime	Just-In-Time debugger	15.0.27005.2	Recommended
Microsoft.VisualStudio.Component.EntityFramework	Entity Framework 6 tools	15.6.27406.0	Recommended
Component.Dotfuscator	PreEmptive Protection - Dotfuscator	15.0.26208.0	Optional
Component.Microsoft.VisualStudio.RazorExtension	Razor Language Services	15.0.26720.2	Optional
Component.Microsoft.Web.LibraryManager	Library Manager	15.8.27705.0	Optional
Component.WebSocket	WebSocket4Net	15.0.26606.0	Optional
Microsoft.Net.Component.4.6.2.SDK	.NET Framework 4.6.2 SDK	15.6.27406.0	Optional
Microsoft.Net.Component.4.6.2.TargetingPack	.NET Framework 4.6.2 targeting pack	15.6.27406.0	Optional

Component ID	Name	Version	Dependency Type
Microsoft.Net.Component.4.7.1.SDK	.NET Framework 4.7.1 SDK	15.6.27406.0	Optional
Microsoft.Net.Component.4.7.1.TargetingPack	.NET Framework 4.7.1 targeting pack	15.6.27406.0	Optional
Microsoft.Net.Component.4.7.2.SDK	.NET Framework 4.7.2 SDK	15.8.27825.0	Optional
Microsoft.Net.Component.4.7.2.TargetingPack	.NET Framework 4.7.2 targeting pack	15.8.27825.0	Optional
Microsoft.Net.Component.4.7.SDK	.NET Framework 4.7 SDK	15.6.27406.0	Optional
Microsoft.Net.Component.4.7.TargetingPack	.NET Framework 4.7 targeting pack	15.6.27406.0	Optional
Microsoft.Net.ComponentGroup.4.6.2.DeveloperTools	.NET Framework 4.6.2 development tools	15.6.27406.0	Optional
Microsoft.Net.ComponentGroup.4.7.1.DeveloperTools	.NET Framework 4.7.1 development tools	15.6.27406.0	Optional
Microsoft.Net.ComponentGroup.4.7.2.DeveloperTools	.NET Framework 4.7.2 development tools	15.8.27825.0	Optional
Microsoft.Net.ComponentGroup.4.7.DeveloperTools	.NET Framework 4.7 development tools	15.6.27406.0	Optional
Microsoft.Net.Core.Component.SDK	.NET Core 2.0 development tools	15.6.27406.0	Optional
Microsoft.Net.Core.Component.SDK.1x	.NET Core 1.0 - 1.1 development tools	15.6.27406.0	Optional
Microsoft.Net.Core.Component.SDK.2.1	.NET Core 2.1 development tools	15.8.27924.0	Optional
Microsoft.NetCore.ComponentGroup.DevelopmentTools	.NET Core 2.0 development tools	15.8.27729.1	Optional
Microsoft.NetCore.ComponentGroup.DevelopmentTools.2.1	.NET Core 2.1 development tools	15.8.27924.0	Optional
Microsoft.VisualStudio.Component.Common.Azure.Tools	Connectivity and publishing tools	15.9.28107.0	Optional
Microsoft.VisualStudio.Component.DockerTools	Container development tools	15.8.27906.1	Optional

Component ID	Name	Version	Dependency Type
Microsoft.VisualStudio.Component.DockerTools.BuildTools	Container development tools - Build Tools	15.7.27617.1	Optional
Microsoft.VisualStudio.Component.FSharp	F# language support	15.8.27825.0	Optional
Microsoft.VisualStudio.Component.FSharp.Desktop	F# desktop language support	15.8.27825.0	Optional
Microsoft.VisualStudio.Component.IIExpress	IIS Express	15.0.26208.0	Optional
Microsoft.VisualStudio.Component.JavaScript.Diagnostics	JavaScript diagnostics	15.8.27729.1	Optional
Microsoft.VisualStudio.Component.JavaScript.TypeScript	JavaScript and TypeScript language support	15.9.28125.51	Optional
Microsoft.VisualStudio.Component.NuGet	NuGet package manager	15.9.28016.0	Optional
Microsoft.VisualStudio.Component.SQL.ADAL	SQL ADAL runtime	15.6.27406.0	Optional
Microsoft.VisualStudio.Component.SQL.CMDUtils	SQL Server Command Line Utilities	15.0.26208.0	Optional
Microsoft.VisualStudio.Component.SQL.DataSources	Data sources for SQL Server support	15.0.26621.2	Optional
Microsoft.VisualStudio.Component.SQL.LocalDB.Runtime	SQL Server Express 2016 LocalDB	15.7.27617.1	Optional
Microsoft.VisualStudio.Component.SQL.NCLI	SQL Server Native Client	15.0.26208.0	Optional
Microsoft.VisualStudio.Component.SQL.SSDT	SQL Server Data Tools	15.9.28107.0	Optional
Microsoft.VisualStudio.Component.TypeScript.3.1	TypeScript 3.1 SDK	15.0.28218.60	Optional
Microsoft.VisualStudio.Component.Wcf.Tooling	Windows Communication Foundation	15.8.27924.0	Optional
Microsoft.VisualStudio.Component.Web	ASP.NET and web development tools	15.8.27825.0	Optional
Microsoft.VisualStudio.ComponentGroup.Web	ASP.NET and web development tools prerequisites	15.9.28219.51	Optional

Component ID	Name	Version	Dependency Type
Microsoft.VisualStudio.ComponentGroup.WebToolsExtensions	ASP.NET and web development	15.8.27825.0	Optional

Game development with Unity

ID: Microsoft.VisualStudio.Workload.ManagedGame

Description: Create 2D and 3D games with Unity, a powerful cross-platform development environment.

Components included by this workload

Component ID	Name	Version	Dependency Type
Microsoft.Net.Component.3.5DeveloperTools	.NET Framework 3.5 development tools	15.6.27406.0	Required
Microsoft.Net.Component.4.7.1.TargetingPack	.NET Framework 4.7.1 targeting pack	15.6.27406.0	Required
Microsoft.VisualStudio.Component.NuGet	NuGet package manager	15.9.28016.0	Required
Microsoft.VisualStudio.Component.Roslyn.Compiler	C# and Visual Basic Roslyn compilers	15.6.27309.0	Required
Microsoft.VisualStudio.Component.Roslyn.LanguageServices	C# and Visual Basic	15.8.27729.1	Required
Microsoft.VisualStudio.Component.Static.Analysis.Tools	Static analysis tools	15.0.26208.0	Required
Microsoft.VisualStudio.Component.Unity	Visual Studio Tools for Unity	15.7.27617.1	Required
Component.UnityEngine.x64	Unity 2018.3 64-bit Editor	15.9.28307.271	Recommended
Component.UnityEngine.x86	Unity 5.6 32-bit Editor	15.6.27406.0	Recommended

Linux development with C++

ID: Microsoft.VisualStudio.Workload.NativeCrossPlat

Description: Create and debug applications running in a Linux environment.

Components included by this workload

Component ID	Name	Version	Dependency Type
Component.MDD.Linux	Visual C++ for Linux Development	15.6.27406.0	Required

Component ID	Name	Version	Dependency Type
Microsoft.VisualStudio.Component.VC.CoreLDE	Visual Studio C++ core features	15.6.27406.0	Required
Microsoft.VisualStudio.Component.Windows10SDK	Windows Universal C Runtime	15.6.27406.0	Required
Component.Linux.CMake	Visual C++ tools for CMake and Linux	15.9.28307.102	Recommended
Microsoft.VisualStudio.Component.Static.Analysis.Tools	Static analysis tools	15.0.26208.0	Recommended
Microsoft.VisualStudio.Component.VC.Tools.x86.x64	VC++ 2017 version 15.9 v14.16 latest v141 tools	15.9.28230.55	Recommended
Microsoft.VisualStudio.Component.Windows10SDK.17763	Windows 10 SDK (10.0.17763.0)	15.9.28307.102	Recommended
Microsoft.VisualStudio.ComponentGroup.WebToolsExtensions	ASP.NET and web development	15.8.27825.0	Recommended
Component.MDD.Linux.GCC.arm	Embedded and IoT Development	15.6.27309.0	Optional

Desktop development with C++

ID: Microsoft.VisualStudio.Workload.NativeDesktop

Description: Build Windows desktop applications using the Microsoft C++ toolset, ATL, or MFC.

Components included by this workload

COMPONENT ID	NAME	VERSION	DEPENDENCY TYPE
Microsoft.Component_MSBuild	MSBuild	15.7.27520.0	Required
Microsoft.VisualStudio.Component.Roslyn.Compiler	C# and Visual Basic Roslyn compilers	15.6.27309.0	Required
Microsoft.VisualStudio.Component.TextTemplating	Text Template Transformation	15.0.26208.0	Required
Microsoft.VisualStudio.Component.VC.CoreLDE	Visual Studio C++ core features	15.6.27406.0	Required
Microsoft.VisualStudio.Component.VC.Redist.14.Latest	Visual C++ 2017 Redistributable Update	15.6.27406.0	Required
Microsoft.VisualStudio.ComponentGroup.NativeDesktop.Core	Visual C++ core desktop features	15.8.27729.1	Required

Component ID	Name	Version	Dependency Type
Microsoft.VisualStudio.Component.Debugger.JustInTime	Just-In-Time debugger	15.0.27005.2	Recommended
Microsoft.VisualStudio.Component.Graphics.Tools	Graphics debugger and GPU profiler for DirectX	15.6.27406.0	Recommended
Microsoft.VisualStudio.Component.Graphics.Win81	Graphics Tools Windows 8.1 SDK	15.6.27406.0	Recommended
Microsoft.VisualStudio.Component.NuGet	NuGet package manager	15.9.28016.0	Recommended
Microsoft.VisualStudio.Component.Static.Analysis.Tools	Static analysis tools	15.0.26208.0	Recommended
Microsoft.VisualStudio.Component.VC.ATL	Visual C++ ATL for x86 and x64	15.7.27625.0	Recommended
Microsoft.VisualStudio.Component.VC.CMake.Project	Visual C++ tools for CMake	15.9.28307.102	Recommended
Microsoft.VisualStudio.Component.VC.DiagnosticTools	C++ profiling tools	15.0.26823.1	Recommended
Microsoft.VisualStudio.Component.VC.TestAdapterForBoostTest	Test Adapter for Boost.Test	15.8.27906.1	Recommended
Microsoft.VisualStudio.Component.VC.TestAdapterForGoogleTest	Test Adapter for Google Test	15.8.27906.1	Recommended
Microsoft.VisualStudio.Component.VC.Tools.x86.x64	VC++ 2017 version 15.9 v14.16 latest v141 tools	15.9.28230.55	Recommended
Microsoft.VisualStudio.Component.Windows10SDK.17763	Windows 10 SDK (10.0.17763.0)	15.9.28307.102	Recommended
Microsoft.VisualStudio.ComponentGroup.WebToolsExtensions	ASP.NET and web development	15.8.27825.0	Recommended
Component.Incredibuild	Incredibuild - Build Acceleration	15.7.27617.1	Optional
Component.IncredibuildMenu	IncredibuildMenu	1.5.0.2	Optional
Microsoft.Component.VC.Runtime.UCRTSDK	Windows Universal CRT SDK	15.6.27309.0	Optional
Microsoft.Net.Component.4.6.1.SDK	.NET Framework 4.6.1 SDK	15.6.27406.0	Optional

Component ID	Name	Version	Dependency Type
Microsoft.Net.Component.4.6.1.TargetingPack	.NET Framework 4.6.1 targeting pack	15.6.27406.0	Optional
Microsoft.VisualStudio.Component.VC.140	VC++ 2015.3 v14.00 (v140) toolset for desktop	15.7.27617.1	Optional
Microsoft.VisualStudio.Component.VC.ATLMFC	Visual C++ MFC for x86 and x64	15.7.27625.0	Optional
Microsoft.VisualStudio.Component.VC.CLI.Support	C++/CLI support	15.6.27309.0	Optional
Microsoft.VisualStudio.Component.VC.Modules.x86.x64	Modules for Standard Library (experimental)	15.6.27309.0	Optional
Microsoft.VisualStudio.Component.Windows10SDK.10240	Windows 10 SDK (10.0.10240.0)	15.6.27406.0	Optional
Microsoft.VisualStudio.Component.Windows10SDK.10586	Windows 10 SDK (10.0.10586.0)	15.6.27406.0	Optional
Microsoft.VisualStudio.Component.Windows10SDK.14393	Windows 10 SDK (10.0.14393.0)	15.6.27406.0	Optional
Microsoft.VisualStudio.Component.Windows10SDK.15063/Desktop	Windows 10 SDK (10.0.15063.0) for Desktop C++ [x86 and x64]	15.6.27406.0	Optional
Microsoft.VisualStudio.Component.Windows10SDK.15063/UWP	Windows 10 SDK (10.0.15063.0) for UWP: C#, VB, JS	15.6.27406.0	Optional
Microsoft.VisualStudio.Component.Windows10SDK.15063/UWP.Native	Windows 10 SDK (10.0.15063.0) for UWP: C++	15.6.27406.0	Optional
Microsoft.VisualStudio.Component.Windows10SDK.16299/Desktop	Windows 10 SDK (10.0.16299.0) for Desktop C++ [x86 and x64]	15.6.27406.0	Optional
Microsoft.VisualStudio.Component.Windows10SDK.16299/Desktop.arm	Windows 10 SDK (10.0.16299.0) for Desktop C++ [ARM and ARM64]	15.6.27406.0	Optional
Microsoft.VisualStudio.Component.Windows10SDK.16299/UWP	Windows 10 SDK (10.0.16299.0) for UWP: C#, VB, JS	15.6.27406.0	Optional
Microsoft.VisualStudio.Component.Windows10SDK.16299/UWP.Native	Windows 10 SDK (10.0.16299.0) for UWP: C++	15.6.27406.0	Optional

Component ID	Name	Version	Dependency Type
Microsoft.VisualStudio.Component.Windows10SDK.17134	Windows 10 SDK (10.0.17134.0)	15.9.28307.102	Optional
Microsoft.VisualStudio.Component.Windows81SDK	Windows 8.1 SDK	15.6.27406.0	Optional
Microsoft.VisualStudio.Component.WinXP	Windows XP support for C++	15.8.27924.0	Optional
Microsoft.VisualStudio.ComponentGroup.NativeDesktop.Win81	Windows 8.1 SDK and UCRT SDK	15.6.27406.0	Optional
Microsoft.VisualStudio.ComponentGroup.NativeDesktop.WinXP	Windows XP support for C++	15.8.27705.0	Optional
Microsoft.VisualStudio.ComponentGroup.Windows10SDK.15063	Windows 10 SDK (10.0.15063.0)	15.8.27825.0	Optional
Microsoft.VisualStudio.ComponentGroup.Windows10SDK.16299	Windows 10 SDK (10.0.16299.0)	15.8.27825.0	Optional

Game development with C++

ID: Microsoft.VisualStudio.Workload.NativeGame

Description: Use the full power of C++ to build professional games powered by DirectX, Unreal, or Cocos2d.

Components included by this workload

Component ID	Name	Version	Dependency Type
Microsoft.VisualStudio.Component.Static.Analysis.Tools	Static analysis tools	15.0.26208.0	Required
Microsoft.VisualStudio.Component.VC.Corelde	Visual Studio C++ core features	15.6.27406.0	Required
Microsoft.VisualStudio.Component.VC.Redist.14.Latest	Visual C++ 2017 Redistributable Update	15.6.27406.0	Required
Microsoft.VisualStudio.Component.VC.Tools.x86.x64	VC++ 2017 version 15.9 v14.16 latest v141 tools	15.9.28230.55	Required
Microsoft.VisualStudio.Component.Windows10SDK	Windows Universal C Runtime	15.6.27406.0	Required
Microsoft.VisualStudio.Component.Graphics.Tools	Graphics debugger and GPU profiler for DirectX	15.6.27406.0	Recommended

Component ID	Name	Version	Dependency Type
Microsoft.VisualStudio.Component.Graphics.Win81	Graphics Tools Windows 8.1 SDK	15.6.27406.0	Recommended
Microsoft.VisualStudio.Component.VC.DiagnosticTools	C++ profiling tools	15.0.26823.1	Recommended
Microsoft.VisualStudio.Component.Windows10SDK.17763	Windows 10 SDK (10.0.17763.0)	15.9.28307.102	Recommended
Component.Android.NDK.R12B	Android NDK (R12B)	12.1.10	Optional
Component.Android.SDK23.Private	Android SDK setup (API level 23) (local install for Mobile development with JavaScript / C++)	15.9.28016.0	Optional
Component.Ant	Apache Ant (1.9.3)	1.9.3.8	Optional
Component.Cocos	Cocos	15.0.26906.1	Optional
Component.Incredibuild	Incredibuild - Build Acceleration	15.7.27617.1	Optional
Component.IncredibuildMenu	IncredibuildMenu	1.5.0.2	Optional
Component.MDD.Android	C++ Android development tools	15.0.26606.0	Optional
Component.OpenJDK	Microsoft distribution OpenJDK	15.9.28125.51	Optional
Component.Unreal	Unreal Engine installer	15.8.27729.1	Optional
Component.Unreal.Android	Visual Studio Android support for Unreal Engine	15.9.28307.341	Optional
Microsoft.Component.VC_Runtime.UCRTSDK	Windows Universal CRT SDK	15.6.27309.0	Optional
Microsoft.Net.Component.4.5.1.TargetingPack	.NET Framework 4.5.1 targeting pack	15.6.27406.0	Optional
Microsoft.Net.Component.4.5.2.TargetingPack	.NET Framework 4.5.2 targeting pack	15.6.27406.0	Optional
Microsoft.Net.Component.4.5.TargetingPack	.NET Framework 4.5 targeting pack	15.6.27406.0	Optional
Microsoft.Net.Component.4.6.1.SDK	.NET Framework 4.6.1 SDK	15.6.27406.0	Optional

Component ID	Name	Version	Dependency Type
Microsoft.Net.Component.4.6.1.TargetingPack	.NET Framework 4.6.1 targeting pack	15.6.27406.0	Optional
Microsoft.Net.Component.4.6.TargetingPack	.NET Framework 4.6 targeting pack	15.6.27406.0	Optional
Microsoft.Net.Component.4.TargetingPack	.NET Framework 4 targeting pack	15.6.27406.0	Optional
Microsoft.Net.ComponentGroup.DevelopmentPrerequisites	.NET Framework 4.6.1 development tools	15.8.27825.0	Optional
Microsoft.Net.ComponentGroup.TargetingPacks.Common	.NET Framework 4 – 4.6 development tools	15.6.27406.0	Optional
Microsoft.VisualStudio.Component.NuGet.BuildTools	NuGet targets and build tasks	15.9.28016.0	Optional
Microsoft.VisualStudio.Component.Roslyn.Compiler	C# and Visual Basic Roslyn compilers	15.6.27309.0	Optional
Microsoft.VisualStudio.Component.Roslyn.LanguageServices	C# and Visual Basic	15.8.27729.1	Optional
Microsoft.VisualStudio.Component.Windows10SDK.10240	Windows 10 SDK (10.0.10240.0)	15.6.27406.0	Optional
Microsoft.VisualStudio.Component.Windows10SDK.10586	Windows 10 SDK (10.0.10586.0)	15.6.27406.0	Optional
Microsoft.VisualStudio.Component.Windows10SDK.14393	Windows 10 SDK (10.0.14393.0)	15.6.27406.0	Optional
Microsoft.VisualStudio.Component.Windows10SDK.15063.Desktop	Windows 10 SDK (10.0.15063.0) for Desktop C++ [x86 and x64]	15.6.27406.0	Optional
Microsoft.VisualStudio.Component.Windows10SDK.15063.UWP	Windows 10 SDK (10.0.15063.0) for UWP: C#, VB, JS	15.6.27406.0	Optional
Microsoft.VisualStudio.Component.Windows10SDK.15063.UWP.Native	Windows 10 SDK (10.0.15063.0) for UWP: C++	15.6.27406.0	Optional
Microsoft.VisualStudio.Component.Windows10SDK.16299.Desktop	Windows 10 SDK (10.0.16299.0) for Desktop C++ [x86 and x64]	15.6.27406.0	Optional

Component ID	Name	Version	Dependency Type
Microsoft.VisualStudio.Component.Windows10SDK.16299.9.Desktop.arm	Windows 10 SDK (10.0.16299.0) for Desktop C++ [ARM and ARM64]	15.6.27406.0	Optional
Microsoft.VisualStudio.Component.Windows10SDK.16299.UWP	Windows 10 SDK (10.0.16299.0) for UWP: C#, VB, JS	15.6.27406.0	Optional
Microsoft.VisualStudio.Component.Windows10SDK.16299.UWP.Native	Windows 10 SDK (10.0.16299.0) for UWP: C++	15.6.27406.0	Optional
Microsoft.VisualStudio.Component.Windows10SDK.17134	Windows 10 SDK (10.0.17134.0)	15.9.28307.102	Optional
Microsoft.VisualStudio.Component.Windows81SDK	Windows 8.1 SDK	15.6.27406.0	Optional
Microsoft.VisualStudio.ComponentGroup.NativeDesktop.Win81	Windows 8.1 SDK and UCRT SDK	15.6.27406.0	Optional
Microsoft.VisualStudio.ComponentGroup.Windows10SDK.15063	Windows 10 SDK (10.0.15063.0)	15.8.27825.0	Optional
Microsoft.VisualStudio.ComponentGroup.Windows10SDK.16299	Windows 10 SDK (10.0.16299.0)	15.8.27825.0	Optional

Mobile development with C++

ID: Microsoft.VisualStudio.Workload.NativeMobile

Description: Build cross-platform applications for iOS, Android or Windows using C++.

Components included by this workload

Component ID	Name	Version	Dependency Type
Component.Android.SDK19.Private	Android SDK setup (API level 19) (local install for Mobile development with JavaScript / C++)	15.9.28107.0	Required
Component.Android.SDK21.Private	Android SDK setup (API level 21) (local install for Mobile development with JavaScript / C++)	15.9.28016.0	Required
Component.Android.SDK22.Private	Android SDK setup (API level 22) (local install for Mobile development with JavaScript / C++)	15.9.28016.0	Required

Component ID	Name	Version	Dependency Type
Component.Android.SDK23.Private	Android SDK setup (API level 23) (local install for Mobile development with JavaScript / C++)	15.9.28016.0	Required
Component.Android.SDK25.Private	Android SDK setup (API level 25) (local install for Mobile development with JavaScript / C++)	15.9.28016.0	Required
Component.OpenJDK	Microsoft distribution OpenJDK	15.9.28125.51	Required
Microsoft.VisualStudio.Component.VC.Corede	Visual Studio C++ core features	15.6.27406.0	Required
Component.Android.NDK.R15C	Android NDK (R15C)	15.2.1	Recommended
Component.Ant	Apache Ant (1.9.3)	1.9.3.8	Recommended
Component.MDD.Android	C++ Android development tools	15.0.26606.0	Recommended
Component.Android.NDK.R12B	Android NDK (R12B)	12.1.10	Optional
Component.Android.NDK.R12B_3264	Android NDK (R12B) (32bit)	12.1.11	Optional
Component.Android.NDK.R13B	Android NDK (R13B)	13.1.7	Optional
Component.Android.NDK.R13B_3264	Android NDK (R13B) (32bit)	13.1.8	Optional
Component.Android.NDK.R15C_3264	Android NDK (R15C) (32bit)	15.2.1	Optional
Component.Google.Android.Emulator.API23.Private	Google Android Emulator (API Level 23) (local install)	15.6.27413.0	Optional
Component.HAXM.Private	Intel Hardware Accelerated Execution Manager (HAXM) (local install)	15.9.28307.421	Optional
Component.Incredibuild	Incredibuild - Build Acceleration	15.7.27617.1	Optional
Component.IncredibuildMenu	IncredibuildMenu	1.5.0.2	Optional
Component.MDD.IOS	C++ iOS development tools	15.0.26621.2	Optional

.NET Core cross-platform development

ID: Microsoft.VisualStudio.Workload.NetCoreTools

Description: Build cross-platform applications using .NET Core, ASP.NET Core, HTML/JavaScript, and Containers including Docker support.

Components included by this workload

Component ID	Name	Version	Dependency Type
Component.Microsoft.VisualStudio.RazorExtension	Razor Language Services	15.0.26720.2	Required
Component.Microsoft.Web.LibraryManager	Library Manager	15.8.27705.0	Required
Component.WebSocket	WebSocket4Net	15.0.26606.0	Required
Microsoft.Component.ClickOnce	ClickOnce Publishing	15.8.27825.0	Required
Microsoft.Component.MSBuild	MSBuild	15.7.27520.0	Required
Microsoft.Net.Component.4.5.2.TargetingPack	.NET Framework 4.5.2 targeting pack	15.6.27406.0	Required
Microsoft.Net.Component.4.5.TargetingPack	.NET Framework 4.5 targeting pack	15.6.27406.0	Required
Microsoft.Net.Component.4.6.1.SDK	.NET Framework 4.6.1 SDK	15.6.27406.0	Required
Microsoft.Net.Component.4.6.1.TargetingPack	.NET Framework 4.6.1 targeting pack	15.6.27406.0	Required
Microsoft.Net.ComponentGroup.DevelopmentPrerequisites	.NET Framework 4.6.1 development tools	15.8.27825.0	Required
Microsoft.Net.Core.Component.SDK.2.1	.NET Core 2.1 development tools	15.8.27924.0	Required
Microsoft.NetCore.ComponentGroup.DevelopmentTools.2.1	.NET Core 2.1 development tools	15.8.27924.0	Required
Microsoft.NetCore.ComponentGroup.Web.2.1	.NET Core 2.1 development tools	15.8.27924.0	Required
Microsoft.VisualStudio.Component.Common.Azure.Tools	Connectivity and publishing tools	15.9.28107.0	Required
Microsoft.VisualStudio.Component.DockerTools	Container development tools	15.8.27906.1	Required

Component ID	Name	Version	Dependency Type
Microsoft.VisualStudio.Component.DockerTools.BuildTools	Container development tools - Build Tools	15.7.27617.1	Required
Microsoft.VisualStudio.Component.FSharp	F# language support	15.8.27825.0	Required
Microsoft.VisualStudio.Component.FSharp.WebTemplates	F# language support for web projects	15.9.28307.421	Required
Microsoft.VisualStudio.Component.IIExpress	IIS Express	15.0.26208.0	Required
Microsoft.VisualStudio.Component.JavaScript.Diagnostics	JavaScript diagnostics	15.8.27729.1	Required
Microsoft.VisualStudio.Component.JavaScript.TypeScript	JavaScript and TypeScript language support	15.9.28125.51	Required
Microsoft.VisualStudio.Component.ManagedDesktop.Core	Managed Desktop Workload Core	15.8.27729.1	Required
Microsoft.VisualStudio.Component.NuGet	NuGet package manager	15.9.28016.0	Required
Microsoft.VisualStudio.Component.PortableLibrary	.NET Portable Library targeting pack	15.6.27309.0	Required
Microsoft.VisualStudio.Component.Roslyn.Compiler	C# and Visual Basic Roslyn compilers	15.6.27309.0	Required
Microsoft.VisualStudio.Component.Roslyn.LanguageServices	C# and Visual Basic	15.8.27729.1	Required
Microsoft.VisualStudio.Component.SQL.ADAL	SQL ADAL runtime	15.6.27406.0	Required
Microsoft.VisualStudio.Component.SQL.CLR	CLR data types for SQL Server	15.0.26208.0	Required
Microsoft.VisualStudio.Component.SQL.CMDUtils	SQL Server Command Line Utilities	15.0.26208.0	Required
Microsoft.VisualStudio.Component.SQL.DataSources	Data sources for SQL Server support	15.0.26621.2	Required
Microsoft.VisualStudio.Component.SQL.LocalDB.Runtime	SQL Server Express 2016 LocalDB	15.7.27617.1	Required
Microsoft.VisualStudio.Component.SQL.NCLI	SQL Server Native Client	15.0.26208.0	Required

Component ID	Name	Version	Dependency Type
Microsoft.VisualStudio.Component.SQL.SSDT	SQL Server Data Tools	15.9.28107.0	Required
Microsoft.VisualStudio.Component.Static.Analysis.Tools	Static analysis tools	15.0.26208.0	Required
Microsoft.VisualStudio.Component.TextTemplating	Text Template Transformation	15.0.26208.0	Required
Microsoft.VisualStudio.Component.TypeScript.3.1	TypeScript 3.1 SDK	15.0.28218.60	Required
Microsoft.VisualStudio.Component.VisualStudioData	Data sources and service references	15.6.27406.0	Required
Microsoft.VisualStudio.ComponentGroup.Web	ASP.NET and web development tools prerequisites	15.9.28219.51	Required
Microsoft.VisualStudio.ComponentGroup.WebToolsExtensions	ASP.NET and web development	15.8.27825.0	Required
Component.Microsoft.VisualStudio.Web.AzureFunctions	Microsoft Azure WebJobs Tools	15.7.27617.1	Recommended
Microsoft.VisualStudio.Component.AppInsights.Tools	Developer Analytics tools	15.8.27825.0	Recommended
Microsoft.VisualStudio.Component.Azure.AuthoringTools	Azure Authoring Tools	15.9.28307.421	Recommended
Microsoft.VisualStudio.Component.Azure.ClientLibs	Azure libraries for .NET	15.0.26208.0	Recommended
Microsoft.VisualStudio.Component.Azure.Compute.Emulator	Azure Compute Emulator	15.9.28307.421	Recommended
Microsoft.VisualStudio.Component.Azure.Storage.Emulator	Azure Storage Emulator	15.9.28125.51	Recommended
Microsoft.VisualStudio.Component.CloudExplorer	Cloud Explorer	15.9.28230.55	Recommended
Microsoft.VisualStudio.Component.DiagnosticTools	.NET profiling tools	15.8.27729.1	Recommended
Microsoft.VisualStudio.Component.Web	ASP.NET and web development tools	15.8.27825.0	Recommended
Microsoft.VisualStudio.Component.WebDeploy	Web Deploy	15.8.27729.1	Recommended

COMPONENT ID	NAME	VERSION	DEPENDENCY TYPE
Microsoft.VisualStudio.ComponentGroup.AzureFunctions	Microsoft Azure WebJobs Tools	15.7.27617.1	Recommended
Microsoft.VisualStudio.ComponentGroup.Web.CloudTools	Cloud tools for web development	15.8.27729.1	Recommended
Microsoft.Net.Core.Component.SDK	.NET Core 2.0 development tools	15.6.27406.0	Optional
Microsoft.Net.Core.Component.SDK.1x	.NET Core 1.0 - 1.1 development tools	15.6.27406.0	Optional
Microsoft.NetCore.1x.ComponentGroup.Web	.NET Core 1.0 - 1.1 development tools for Web	15.6.27406.0	Optional
Microsoft.NetCore.ComponentGroup.DevelopmentTools	.NET Core 2.0 development tools	15.8.27729.1	Optional
Microsoft.NetCore.ComponentGroup.Web	.NET Core 2.0 development tools	15.7.27625.0	Optional
Microsoft.VisualStudio.ComponentGroup.IISDevelopment	Development time IIS support	15.9.28219.51	Optional

Mobile development with .NET

ID: Microsoft.VisualStudio.Workload.NetCrossPlat

Description: Build cross-platform applications for iOS, Android or Windows using Xamarin.

Components included by this workload

COMPONENT ID	NAME	VERSION	DEPENDENCY TYPE
Component.Xamarin	Xamarin	15.8.27906.1	Required
Component.Xamarin.RemotedSimulator	Xamarin Remoted Simulator	15.6.27323.2	Required
Microsoft.Component.MSBuild	MSBuild	15.7.27520.0	Required
Microsoft.Net.Component.4.6.1.SDK	.NET Framework 4.6.1 SDK	15.6.27406.0	Required
Microsoft.Net.Component.4.6.1.TargetingPack	.NET Framework 4.6.1 targeting pack	15.6.27406.0	Required
Microsoft.Net.ComponentGroup.DevelopmentPrerequisites	.NET Framework 4.6.1 development tools	15.8.27825.0	Required
Microsoft.Net.Core.Component.SDK	.NET Core 2.0 development tools	15.6.27406.0	Required

Component ID	Name	Version	Dependency Type
Microsoft.NetCore.ComponentGroup.DevelopmentTools	.NET Core 2.0 development tools	15.8.27729.1	Required
Microsoft.VisualStudio.Component.FSharp	F# language support	15.8.27825.0	Required
Microsoft.VisualStudio.Component.Merq	Common Xamarin internal tools	15.8.27924.0	Required
Microsoft.VisualStudio.Component.MonoDebugger	Mono debugger	15.0.26720.2	Required
Microsoft.VisualStudio.Component.NuGet	NuGet package manager	15.9.28016.0	Required
Microsoft.VisualStudio.Component.PortableLibrary	.NET Portable Library targeting pack	15.6.27309.0	Required
Microsoft.VisualStudio.Component.Roslyn.Compiler	C# and Visual Basic Roslyn compilers	15.6.27309.0	Required
Microsoft.VisualStudio.Component.Roslyn.LanguageServices	C# and Visual Basic	15.8.27729.1	Required
Microsoft.VisualStudio.Component.Static.Analysis.Tools	Static analysis tools	15.0.26208.0	Required
Microsoft.VisualStudio.ComponentGroup.WebToolsExtensions.TemplateEngine	ASP.NET templating engine	15.8.27729.1	Required
Component.Android.SDK27	Android SDK setup (API level 27)	15.9.28016.0	Recommended
Component.Google.Android.Emulator.API27	Google Android Emulator (API Level 27)	15.9.28307.421	Recommended
Component.HAXM	Intel Hardware Accelerated Execution Manager (HAXM) (global install)	15.9.28307.421	Recommended
Component.OpenJDK	Microsoft distribution OpenJDK	15.9.28125.51	Recommended
Component.Xamarin.Inspect	Xamarin Workbooks	15.0.26606.0	Optional
Microsoft.Component.ClickOnce	ClickOnce Publishing	15.8.27825.0	Optional
Microsoft.Component.NetFX.Native	.NET Native	15.0.26208.0	Optional

Component ID	Name	Version	Dependency Type
Microsoft.VisualStudio.Component.AppInsights.Tools	Developer Analytics tools	15.8.27825.0	Optional
Microsoft.VisualStudio.Component.DiagnosticTools	.NET profiling tools	15.8.27729.1	Optional
Microsoft.VisualStudio.Component.Graphics	Image and 3D model editors	15.6.27406.0	Optional
Microsoft.VisualStudio.Component.SQL.CLR	CLR data types for SQL Server	15.0.26208.0	Optional
Microsoft.VisualStudio.Component.VisualStudioData	Data sources and service references	15.6.27406.0	Optional
Microsoft.VisualStudio.Component.Windows10SDK.17763	Windows 10 SDK (10.0.17763.0)	15.9.28307.102	Optional
Microsoft.VisualStudio.ComponentGroup.UWP.Xamarin	Universal Windows Platform tools for Xamarin	15.9.28307.102	Optional

ASP.NET and web development

ID: Microsoft.VisualStudio.Workload.NetWeb

Description: Build web applications using ASP.NET, ASP.NET Core, HTML/JavaScript, and Containers including Docker support.

Components included by this workload

Component ID	Name	Version	Dependency Type
Component.Microsoft.VisualStudio.RazorExtension	Razor Language Services	15.0.26720.2	Required
Component.Microsoft.Web.LibraryManager	Library Manager	15.8.27705.0	Required
Component.WebSocket	WebSocket4Net	15.0.26606.0	Required
Microsoft.Component.ClickOnce	ClickOnce Publishing	15.8.27825.0	Required
Microsoft.Component.MSBuild	MSBuild	15.7.27520.0	Required
Microsoft.Net.Component.4.5.2.TargetingPack	.NET Framework 4.5.2 targeting pack	15.6.27406.0	Required
Microsoft.Net.Component.4.5.TargetingPack	.NET Framework 4.5 targeting pack	15.6.27406.0	Required

Component ID	Name	Version	Dependency Type
Microsoft.Net.Component.4.6.1.SDK	.NET Framework 4.6.1 SDK	15.6.27406.0	Required
Microsoft.Net.Component.4.6.1.TargetingPack	.NET Framework 4.6.1 targeting pack	15.6.27406.0	Required
Microsoft.Net.ComponentGroup.DevelopmentPrerequisites	.NET Framework 4.6.1 development tools	15.8.27825.0	Required
Microsoft.Net.Core.Component.SDK.2.1	.NET Core 2.1 development tools	15.8.27924.0	Required
Microsoft.NetCore.ComponentGroup.DevelopmentTools.2.1	.NET Core 2.1 development tools	15.8.27924.0	Required
Microsoft.NetCore.ComponentGroup.Web.2.1	.NET Core 2.1 development tools	15.8.27924.0	Required
Microsoft.VisualStudio.Component.Common.Azure.Tools	Connectivity and publishing tools	15.9.28107.0	Required
Microsoft.VisualStudio.Component.DockerTools	Container development tools	15.8.27906.1	Required
Microsoft.VisualStudio.Component.DockerTools.BuildTools	Container development tools - Build Tools	15.7.27617.1	Required
Microsoft.VisualStudio.Component.FSharp	F# language support	15.8.27825.0	Required
Microsoft.VisualStudio.Component.FSharp.WebTemplates	F# language support for web projects	15.9.28307.421	Required
Microsoft.VisualStudio.Component.IISExpress	IIS Express	15.0.26208.0	Required
Microsoft.VisualStudio.Component.JavaScript.Diagnostics	JavaScript diagnostics	15.8.27729.1	Required
Microsoft.VisualStudio.Component.JavaScript.TypeScript	JavaScript and TypeScript language support	15.9.28125.51	Required
Microsoft.VisualStudio.Component.ManagedDesktop.Core	Managed Desktop Workload Core	15.8.27729.1	Required
Microsoft.VisualStudio.Component.NuGet	NuGet package manager	15.9.28016.0	Required
Microsoft.VisualStudio.Component.PortableLibrary	.NET Portable Library targeting pack	15.6.27309.0	Required

Component ID	Name	Version	Dependency Type
Microsoft.VisualStudio.Component.Roslyn.Compiler	C# and Visual Basic Roslyn compilers	15.6.27309.0	Required
Microsoft.VisualStudio.Component.Roslyn.LanguageServices	C# and Visual Basic	15.8.27729.1	Required
Microsoft.VisualStudio.Component.SQL.ADAL	SQL ADAL runtime	15.6.27406.0	Required
Microsoft.VisualStudio.Component.SQL.CLR	CLR data types for SQL Server	15.0.26208.0	Required
Microsoft.VisualStudio.Component.SQL.CMDUtils	SQL Server Command Line Utilities	15.0.26208.0	Required
Microsoft.VisualStudio.Component.SQL.DataSources	Data sources for SQL Server support	15.0.26621.2	Required
Microsoft.VisualStudio.Component.SQL.LocalDB.Runtime	SQL Server Express 2016 LocalDB	15.7.27617.1	Required
Microsoft.VisualStudio.Component.SQL.NCLI	SQL Server Native Client	15.0.26208.0	Required
Microsoft.VisualStudio.Component.SQL.SSDT	SQL Server Data Tools	15.9.28107.0	Required
Microsoft.VisualStudio.Component.Static.Analysis.Tools	Static analysis tools	15.0.26208.0	Required
Microsoft.VisualStudio.Component.TextTemplating	Text Template Transformation	15.0.26208.0	Required
Microsoft.VisualStudio.Component.TypeScript.3.1	TypeScript 3.1 SDK	15.0.28218.60	Required
Microsoft.VisualStudio.Component.VisualStudioData	Data sources and service references	15.6.27406.0	Required
Microsoft.VisualStudio.Component.Web	ASP.NET and web development tools	15.8.27825.0	Required
Microsoft.VisualStudio.ComponentGroup.Web	ASP.NET and web development tools prerequisites	15.9.28219.51	Required
Microsoft.VisualStudio.ComponentGroup.WebToolsExtensions	ASP.NET and web development	15.8.27825.0	Required
Component.Microsoft.VisualStudio.Web.AzureFunctions	Microsoft Azure WebJobs Tools	15.7.27617.1	Recommended

Component ID	Name	Version	Dependency Type
Microsoft.Net.Component.4.5.1.TargetingPack	.NET Framework 4.5.1 targeting pack	15.6.27406.0	Recommended
Microsoft.Net.Component.4.6.TargetingPack	.NET Framework 4.6 targeting pack	15.6.27406.0	Recommended
Microsoft.Net.Component.4.TargetingPack	.NET Framework 4 targeting pack	15.6.27406.0	Recommended
Microsoft.Net.ComponentGroup.TargetingPacks.Common	.NET Framework 4 – 4.6 development tools	15.6.27406.0	Recommended
Microsoft.VisualStudio.Component.AppInsights.Tools	Developer Analytics tools	15.8.27825.0	Recommended
Microsoft.VisualStudio.Component.AspNet45	Advanced ASP.NET features	15.7.27625.0	Recommended
Microsoft.VisualStudio.Component.Azure.AuthoringTools	Azure Authoring Tools	15.9.28307.421	Recommended
Microsoft.VisualStudio.Component.Azure.ClientLibs	Azure libraries for .NET	15.0.26208.0	Recommended
Microsoft.VisualStudio.Component.Azure.Compute Emulator	Azure Compute Emulator	15.9.28307.421	Recommended
Microsoft.VisualStudio.Component.Azure.Storage Emulator	Azure Storage Emulator	15.9.28125.51	Recommended
Microsoft.VisualStudio.Component.CloudExplorer	Cloud Explorer	15.9.28230.55	Recommended
Microsoft.VisualStudio.Component.DiagnosticTools	.NET profiling tools	15.8.27729.1	Recommended
Microsoft.VisualStudio.Component.EntityFramework	Entity Framework 6 tools	15.6.27406.0	Recommended
Microsoft.VisualStudio.Component.Wcf.Tooling	Windows Communication Foundation	15.8.27924.0	Recommended
Microsoft.VisualStudio.Component.WebDeploy	Web Deploy	15.8.27729.1	Recommended
Microsoft.VisualStudio.ComponentGroup.AzureFunctions	Microsoft Azure WebJobs Tools	15.7.27617.1	Recommended
Microsoft.VisualStudio.ComponentGroup.Web.CloudTools	Cloud tools for web development	15.8.27729.1	Recommended

Component ID	Name	Version	Dependency Type
Microsoft.Net.Component.4.6.2.SDK	.NET Framework 4.6.2 SDK	15.6.27406.0	Optional
Microsoft.Net.Component.4.6.2.TargetingPack	.NET Framework 4.6.2 targeting pack	15.6.27406.0	Optional
Microsoft.Net.Component.4.7.1.SDK	.NET Framework 4.7.1 SDK	15.6.27406.0	Optional
Microsoft.Net.Component.4.7.1.TargetingPack	.NET Framework 4.7.1 targeting pack	15.6.27406.0	Optional
Microsoft.Net.Component.4.7.2.SDK	.NET Framework 4.7.2 SDK	15.8.27825.0	Optional
Microsoft.Net.Component.4.7.2.TargetingPack	.NET Framework 4.7.2 targeting pack	15.8.27825.0	Optional
Microsoft.Net.Component.4.7.SDK	.NET Framework 4.7 SDK	15.6.27406.0	Optional
Microsoft.Net.Component.4.7.TargetingPack	.NET Framework 4.7 targeting pack	15.6.27406.0	Optional
Microsoft.Net.ComponentGroup.4.6.2.DeveloperTools	.NET Framework 4.6.2 development tools	15.6.27406.0	Optional
Microsoft.Net.ComponentGroup.4.7.1.DeveloperTools	.NET Framework 4.7.1 development tools	15.6.27406.0	Optional
Microsoft.Net.ComponentGroup.4.7.2.DeveloperTools	.NET Framework 4.7.2 development tools	15.8.27825.0	Optional
Microsoft.Net.ComponentGroup.4.7.DeveloperTools	.NET Framework 4.7 development tools	15.6.27406.0	Optional
Microsoft.Net.Core.Component.SDK	.NET Core 2.0 development tools	15.6.27406.0	Optional
Microsoft.Net.Core.Component.SDK.1x	.NET Core 1.0 - 1.1 development tools	15.6.27406.0	Optional
Microsoft.NetCore.1x.ComponentGroup.Web	.NET Core 1.0 - 1.1 development tools for Web	15.6.27406.0	Optional
Microsoft.NetCore.ComponentGroup.DevelopmentTools	.NET Core 2.0 development tools	15.8.27729.1	Optional
Microsoft.NetCore.ComponentGroup.Web	.NET Core 2.0 development tools	15.7.27625.0	Optional
Microsoft.VisualStudio.ComponentGroup.IISDevelopment	Development time IIS support	15.9.28219.51	Optional

Component ID	Name	Version	Dependency Type
Microsoft.VisualStudio.Web.Mvc4.ComponentGroup	ASP.NET MVC 4	15.6.27406.0	Optional

Node.js development

ID: Microsoft.VisualStudio.Workload.Node

Description: Build scalable network applications using Node.js, an asynchronous event-driven JavaScript runtime.

Components included by this workload

Component ID	Name	Version	Dependency Type
Component.WebSocket	WebSocket4Net	15.0.26606.0	Required
Microsoft.VisualStudio.Component.JavaScript.Diagnostics	JavaScript diagnostics	15.8.27729.1	Required
Microsoft.VisualStudio.Component.JavaScript.TypeScript	JavaScript and TypeScript language support	15.9.28125.51	Required
Microsoft.VisualStudio.Component.Node.Build	Node.js MSBuild support	15.8.27825.0	Required
Microsoft.VisualStudio.Component.Node.Tools	Node.js development support	15.8.27825.0	Required
Microsoft.VisualStudio.Component.NuGet	NuGet package manager	15.9.28016.0	Required
Microsoft.VisualStudio.Component.TestTools.Core	Testing tools core features	15.7.27520.0	Required
Microsoft.VisualStudio.Component.TypeScript.3.1	TypeScript 3.1 SDK	15.0.28218.60	Required
Microsoft.VisualStudio.ComponentGroup.WebToolsExtensions	ASP.NET and web development	15.8.27825.0	Required
Microsoft.VisualStudio.Component.WebDeploy	Web Deploy	15.8.27729.1	Recommended
Microsoft.VisualStudio.Component.AppInsights.Tools	Developer Analytics tools	15.8.27825.0	Optional
Microsoft.VisualStudio.Component.Common.Azure.Tools	Connectivity and publishing tools	15.9.28107.0	Optional
Microsoft.VisualStudio.Component.Static.Analysis.Tools	Static analysis tools	15.0.26208.0	Optional

Component ID	Name	Version	Dependency Type
Microsoft.VisualStudio.Component.VC.CoreRedist	Visual Studio C++ core features	15.6.27406.0	Optional
Microsoft.VisualStudio.Component.VC.Tools.x86.x64	VC++ 2017 version 15.9 v14.16 latest v141 tools	15.9.28230.55	Optional

Office/SharePoint development

ID: Microsoft.VisualStudio.Workload.Office

Description: Create Office and SharePoint add-ins, SharePoint solutions, and VSTO add-ins using C#, VB, and JavaScript.

Components included by this workload

Component ID	Name	Version	Dependency Type
Component.Microsoft.VisualStudio.RazorExtension	Razor Language Services	15.0.26720.2	Required
Component.Microsoft.Web.LibraryManager	Library Manager	15.8.27705.0	Required
Component.WebSocket	WebSocket4Net	15.0.26606.0	Required
Microsoft.Component.ClickOnce	ClickOnce Publishing	15.8.27825.0	Required
Microsoft.Component.MSBuild	MSBuild	15.7.27520.0	Required
Microsoft.Net.Component.4.5.2.TargetingPack	.NET Framework 4.5.2 targeting pack	15.6.27406.0	Required
Microsoft.Net.Component.4.5.TargetingPack	.NET Framework 4.5 targeting pack	15.6.27406.0	Required
Microsoft.Net.Component.4.6.1.SDK	.NET Framework 4.6.1 SDK	15.6.27406.0	Required
Microsoft.Net.Component.4.6.1.TargetingPack	.NET Framework 4.6.1 targeting pack	15.6.27406.0	Required
Microsoft.Net.Component.4.TargetingPack	.NET Framework 4 targeting pack	15.6.27406.0	Required
Microsoft.Net.ComponentGroup.DevelopmentPrerequisites	.NET Framework 4.6.1 development tools	15.8.27825.0	Required
Microsoft.Net.Core.Component.SDK.2.1	.NET Core 2.1 development tools	15.8.27924.0	Required

Component ID	Name	Version	Dependency Type
Microsoft.VisualStudio.Component.AppInsights.Tools	Developer Analytics tools	15.8.27825.0	Required
Microsoft.VisualStudio.Component.Common.Azure.Tools	Connectivity and publishing tools	15.9.28107.0	Required
Microsoft.VisualStudio.Component.DockerTools	Container development tools	15.8.27906.1	Required
Microsoft.VisualStudio.Component.DockerTools.BuildTools	Container development tools - Build Tools	15.7.27617.1	Required
Microsoft.VisualStudio.Component.IISExpress	IIS Express	15.0.26208.0	Required
Microsoft.VisualStudio.Component.JavaScript.Diagnostics	JavaScript diagnostics	15.8.27729.1	Required
Microsoft.VisualStudio.Component.JavaScript.TypeScript	JavaScript and TypeScript language support	15.9.28125.51	Required
Microsoft.VisualStudio.Component.ManagedDesktop.Core	Managed Desktop Workload Core	15.8.27729.1	Required
Microsoft.VisualStudio.Component.ManagedDesktop.Prerequisites	.NET desktop development tools	15.7.27625.0	Required
Microsoft.VisualStudio.Component.NuGet	NuGet package manager	15.9.28016.0	Required
Microsoft.VisualStudio.Component.PortableLibrary	.NET Portable Library targeting pack	15.6.27309.0	Required
Microsoft.VisualStudio.Component.Roslyn.Compiler	C# and Visual Basic Roslyn compilers	15.6.27309.0	Required
Microsoft.VisualStudio.Component.Roslyn.LanguageServices	C# and Visual Basic	15.8.27729.1	Required
Microsoft.VisualStudio.Component.Sharepoint.Tools	Office Developer Tools for Visual Studio	15.8.27924.0	Required
Microsoft.VisualStudio.Component.SQL.ADAL	SQL ADAL runtime	15.6.27406.0	Required
Microsoft.VisualStudio.Component.SQL.CLR	CLR data types for SQL Server	15.0.26208.0	Required
Microsoft.VisualStudio.Component.SQL.CMDUtils	SQL Server Command Line Utilities	15.0.26208.0	Required

Component ID	Name	Version	Dependency Type
Microsoft.VisualStudio.Component.SQL.DataSources	Data sources for SQL Server support	15.0.26621.2	Required
Microsoft.VisualStudio.Component.SQL.LocalDB.Runtime	SQL Server Express 2016 LocalDB	15.7.27617.1	Required
Microsoft.VisualStudio.Component.SQL.NCLI	SQL Server Native Client	15.0.26208.0	Required
Microsoft.VisualStudio.Component.SQL.SSDT	SQL Server Data Tools	15.9.28107.0	Required
Microsoft.VisualStudio.Component.Static.Analysis.Tools	Static analysis tools	15.0.26208.0	Required
Microsoft.VisualStudio.Component.TextTemplating	Text Template Transformation	15.0.26208.0	Required
Microsoft.VisualStudio.Component.TypeScript.3.1	TypeScript 3.1 SDK	15.0.28218.60	Required
Microsoft.VisualStudio.Component.VisualStudioData	Data sources and service references	15.6.27406.0	Required
Microsoft.VisualStudio.Component.Wcf.Tooling	Windows Communication Foundation	15.8.27924.0	Required
Microsoft.VisualStudio.Component.Web	ASP.NET and web development tools	15.8.27825.0	Required
Microsoft.VisualStudio.Component.Workflow	Windows Workflow Foundation	15.8.27825.0	Required
Microsoft.VisualStudio.ComponentGroup.Web	ASP.NET and web development tools prerequisites	15.9.28219.51	Required
Microsoft.VisualStudio.ComponentGroup.WebToolsExtensions	ASP.NET and web development	15.8.27825.0	Required
Microsoft.VisualStudio.Component.TeamOffice	Visual Studio Tools for Office (VSTO)	15.7.27625.0	Recommended
Microsoft.VisualStudio.Component.WebDeploy	Web Deploy	15.8.27729.1	Recommended
Microsoft.Net.Component.4.6.2.SDK	.NET Framework 4.6.2 SDK	15.6.27406.0	Optional
Microsoft.Net.Component.4.6.2.TargetingPack	.NET Framework 4.6.2 targeting pack	15.6.27406.0	Optional

Component ID	Name	Version	Dependency Type
Microsoft.Net.Component.4.7.1.SDK	.NET Framework 4.7.1 SDK	15.6.27406.0	Optional
Microsoft.Net.Component.4.7.1.TargetingPack	.NET Framework 4.7.1 targeting pack	15.6.27406.0	Optional
Microsoft.Net.Component.4.7.2.SDK	.NET Framework 4.7.2 SDK	15.8.27825.0	Optional
Microsoft.Net.Component.4.7.2.TargetingPack	.NET Framework 4.7.2 targeting pack	15.8.27825.0	Optional
Microsoft.Net.Component.4.7.SDK	.NET Framework 4.7 SDK	15.6.27406.0	Optional
Microsoft.Net.Component.4.7.TargetingPack	.NET Framework 4.7 targeting pack	15.6.27406.0	Optional
Microsoft.Net.ComponentGroup.4.6.2.DeveloperTools	.NET Framework 4.6.2 development tools	15.6.27406.0	Optional
Microsoft.Net.ComponentGroup.4.7.1.DeveloperTools	.NET Framework 4.7.1 development tools	15.6.27406.0	Optional
Microsoft.Net.ComponentGroup.4.7.2.DeveloperTools	.NET Framework 4.7.2 development tools	15.8.27825.0	Optional
Microsoft.Net.ComponentGroup.4.7.DeveloperTools	.NET Framework 4.7 development tools	15.6.27406.0	Optional

Python development

ID: Microsoft.VisualStudio.Workload.Python

Description: Editing, debugging, interactive development and source control for Python.

Components included by this workload

Component ID	Name	Version	Dependency Type
Microsoft.Component.PythonTools	Python language support	15.0.26823.1	Required
Component.CPython3.x64	Python 3 64-bit (3.6.6)	3.6.6	Recommended
Microsoft.Component.CookiecutterTools	Cookiecutter template support	15.0.26621.2	Recommended
Microsoft.Component.PythonTools.Web	Python web support	15.9.28107.0	Recommended
Microsoft.VisualStudio.Component.Common.Azure.Tools	Connectivity and publishing tools	15.9.28107.0	Recommended

Component ID	Name	Version	Dependency Type
Microsoft.VisualStudio.Component.JavaScript.TypeScript	JavaScript and TypeScript language support	15.9.28125.51	Recommended
Microsoft.VisualStudio.Component.SQL.CLR	CLR data types for SQL Server	15.0.26208.0	Recommended
Microsoft.VisualStudio.Component.TypeScript.3.1	TypeScript 3.1 SDK	15.0.28218.60	Recommended
Microsoft.VisualStudio.Component.VisualStudioData	Data sources and service references	15.6.27406.0	Recommended
Microsoft.VisualStudio.Component.WebDeploy	Web Deploy	15.8.27729.1	Recommended
Microsoft.VisualStudio.ComponentGroup.WebToolsExtensions	ASP.NET and web development	15.8.27825.0	Recommended
Component.Anaconda2.x64	Anaconda2 64-bit (5.2.0)	5.2.0	Optional
Component.Anaconda2.x86	Anaconda2 32-bit (5.2.0)	5.2.0	Optional
Component.Anaconda3.x64	Anaconda3 64-bit (5.2.0)	5.2.0	Optional
Component.Anaconda3.x86	Anaconda3 32-bit (5.2.0)	5.2.0	Optional
Component.CPython2.x64	Python 2 64-bit (2.7.14)	2.7.14	Optional
Component.CPython2.x86	Python 2 32-bit (2.7.14)	2.7.14	Optional
Component.CPython3.x86	Python 3 32-bit (3.6.6)	3.6.6	Optional
Component.Microsoft.VisualStudio.RazorExtension	Razor Language Services	15.0.26720.2	Optional
Component.Microsoft.Web.LibraryManager	Library Manager	15.8.27705.0	Optional
Component.WebSocket	WebSocket4Net	15.0.26606.0	Optional
Microsoft.Component.ClickOnce	ClickOnce Publishing	15.8.27825.0	Optional
Microsoft.Component_MSBUILD	MSBuild	15.7.27520.0	Optional
Microsoft.Component.NetFX.Native	.NET Native	15.0.26208.0	Optional
Microsoft.Component.PythonTools.UWP	Python IoT support	15.0.26606.0	Optional

Component ID	Name	Version	Dependency Type
Microsoft.Component.VC.Runtime.UCRTSDK	Windows Universal CRT SDK	15.6.27309.0	Optional
Microsoft.ComponentGroup.PythonTools.NativeDevelopment	Python native development tools	15.9.28307.102	Optional
Microsoft.Net.Component.4.5.2.TargetingPack	.NET Framework 4.5.2 targeting pack	15.6.27406.0	Optional
Microsoft.Net.Component.4.5.TargetingPack	.NET Framework 4.5 targeting pack	15.6.27406.0	Optional
Microsoft.Net.Component.4.6.1.SDK	.NET Framework 4.6.1 SDK	15.6.27406.0	Optional
Microsoft.Net.Component.4.6.1.TargetingPack	.NET Framework 4.6.1 targeting pack	15.6.27406.0	Optional
Microsoft.Net.ComponentGroup.DevelopmentPrerequisites	.NET Framework 4.6.1 development tools	15.8.27825.0	Optional
Microsoft.Net.Core.Component.SDK.2.1	.NET Core 2.1 development tools	15.8.27924.0	Optional
Microsoft.VisualStudio.Component.AppInsights.Tools	Developer Analytics tools	15.8.27825.0	Optional
Microsoft.VisualStudio.Component.Azure.AuthoringTools	Azure Authoring Tools	15.9.28307.421	Optional
Microsoft.VisualStudio.Component.Azure.ClientLibs	Azure libraries for .NET	15.0.26208.0	Optional
Microsoft.VisualStudio.Component.Azure.Compute.Emulator	Azure Compute Emulator	15.9.28307.421	Optional
Microsoft.VisualStudio.Component.Azure.Storage.Emulator	Azure Storage Emulator	15.9.28125.51	Optional
Microsoft.VisualStudio.Component.Azure.Waverton	Azure Cloud Services core tools	15.9.28107.0	Optional
Microsoft.VisualStudio.Component.Azure.Waverton.BuildTools	Azure Cloud Services build tools	15.7.27617.1	Optional
Microsoft.VisualStudio.Component.ClassDesigner	Class Designer	15.0.26208.0	Optional

Component ID	Name	Version	Dependency Type
Microsoft.VisualStudio.Component.DiagnosticTools	.NET profiling tools	15.8.27729.1	Optional
Microsoft.VisualStudio.Component.DockerTools	Container development tools	15.8.27906.1	Optional
Microsoft.VisualStudio.Component.DockerTools.BuildTools	Container development tools - Build Tools	15.7.27617.1	Optional
Microsoft.VisualStudio.Component.Graphics	Image and 3D model editors	15.6.27406.0	Optional
Microsoft.VisualStudio.Component.Graphics.Tools	Graphics debugger and GPU profiler for DirectX	15.6.27406.0	Optional
Microsoft.VisualStudio.Component.Graphics.Win81	Graphics Tools Windows 8.1 SDK	15.6.27406.0	Optional
Microsoft.VisualStudio.Component.IISExpress	IIS Express	15.0.26208.0	Optional
Microsoft.VisualStudio.Component.JavaScript.Diagnostics	JavaScript diagnostics	15.8.27729.1	Optional
Microsoft.VisualStudio.Component.ManagedDesktop.Core	Managed Desktop Workload Core	15.8.27729.1	Optional
Microsoft.VisualStudio.Component.NuGet	NuGet package manager	15.9.28016.0	Optional
Microsoft.VisualStudio.Component.PortableLibrary	.NET Portable Library targeting pack	15.6.27309.0	Optional
Microsoft.VisualStudio.Component.Roslyn.Compiler	C# and Visual Basic Roslyn compilers	15.6.27309.0	Optional
Microsoft.VisualStudio.Component.Roslyn.LanguageServices	C# and Visual Basic	15.8.27729.1	Optional
Microsoft.VisualStudio.Component.SQL.ADAL	SQL ADAL runtime	15.6.27406.0	Optional
Microsoft.VisualStudio.Component.SQL.CMDUtils	SQL Server Command Line Utilities	15.0.26208.0	Optional
Microsoft.VisualStudio.Component.SQL.DataSources	Data sources for SQL Server support	15.0.26621.2	Optional
Microsoft.VisualStudio.Component.SQL.LocalDB.Runtime	SQL Server Express 2016 LocalDB	15.7.27617.1	Optional

Component ID	Name	Version	Dependency Type
Microsoft.VisualStudio.Component.SQL.NCLI	SQL Server Native Client	15.0.26208.0	Optional
Microsoft.VisualStudio.Component.SQL.SSDT	SQL Server Data Tools	15.9.28107.0	Optional
Microsoft.VisualStudio.Component.Static.Analysis.Tools	Static analysis tools	15.0.26208.0	Optional
Microsoft.VisualStudio.Component.TextTemplating	Text Template Transformation	15.0.26208.0	Optional
Microsoft.VisualStudio.Component.VC.140	VC++ 2015.3 v14.00 (v140) toolset for desktop	15.7.27617.1	Optional
Microsoft.VisualStudio.Component.VC.CoreLDE	Visual Studio C++ core features	15.6.27406.0	Optional
Microsoft.VisualStudio.Component.VC.DiagnosticTools	C++ profiling tools	15.0.26823.1	Optional
Microsoft.VisualStudio.Component.VC.Tools.x86.x64	VC++ 2017 version 15.9 v14.16 latest v141 tools	15.9.28230.55	Optional
Microsoft.VisualStudio.Component.Web	ASP.NET and web development tools	15.8.27825.0	Optional
Microsoft.VisualStudio.Component.Windows10SDK	Windows Universal C Runtime	15.6.27406.0	Optional
Microsoft.VisualStudio.Component.Windows10SDK.1058.6	Windows 10 SDK (10.0.10586.0)	15.6.27406.0	Optional
Microsoft.VisualStudio.Component.Windows10SDK.1776.3	Windows 10 SDK (10.0.17763.0)	15.9.28307.102	Optional
Microsoft.VisualStudio.Component.Windows81SDK	Windows 8.1 SDK	15.6.27406.0	Optional
Microsoft.VisualStudio.ComponentGroup.Web	ASP.NET and web development tools prerequisites	15.9.28219.51	Optional

Universal Windows Platform development

ID: Microsoft.VisualStudio.Workload.Universal

Description: Create applications for the Universal Windows Platform with C#, VB, JavaScript, or optionally C++.

Components included by this workload

Component ID	Name	Version	Dependency Type
Component.WebSocket	WebSocket4Net	15.0.26606.0	Required
Microsoft.Component.ClickOnce	ClickOnce Publishing	15.8.27825.0	Required
Microsoft.Component.NetFX.Native	.NET Native	15.0.26208.0	Required
Microsoft.ComponentGroup.Blend	Blend for Visual Studio	15.6.27406.0	Required
Microsoft.Net.Component.4.5.TargetingPack	.NET Framework 4.5 targeting pack	15.6.27406.0	Required
Microsoft.Net.Component.4.6.1.SDK	.NET Framework 4.6.1 SDK	15.6.27406.0	Required
Microsoft.Net.Core.Component.SDK.2.1	.NET Core 2.1 development tools	15.8.27924.0	Required
Microsoft.VisualStudio.Component.AppInsights.Tools	Developer Analytics tools	15.8.27825.0	Required
Microsoft.VisualStudio.Component.DiagnosticTools	.NET profiling tools	15.8.27729.1	Required
Microsoft.VisualStudio.Component.Graphics	Image and 3D model editors	15.6.27406.0	Required
Microsoft.VisualStudio.Component.JavaScript.Diagnostics	JavaScript diagnostics	15.8.27729.1	Required
Microsoft.VisualStudio.Component.JavaScript.TypeScript	JavaScript and TypeScript language support	15.9.28125.51	Required
Microsoft.VisualStudio.Component.NuGet	NuGet package manager	15.9.28016.0	Required
Microsoft.VisualStudio.Component.PortableLibrary	.NET Portable Library targeting pack	15.6.27309.0	Required
Microsoft.VisualStudio.Component.Roslyn.Compiler	C# and Visual Basic Roslyn compilers	15.6.27309.0	Required
Microsoft.VisualStudio.Component.Roslyn.LanguageServices	C# and Visual Basic	15.8.27729.1	Required
Microsoft.VisualStudio.Component.SQL.CLR	CLR data types for SQL Server	15.0.26208.0	Required
Microsoft.VisualStudio.Component.Static.Analysis.Tools	Static analysis tools	15.0.26208.0	Required

Component ID	Name	Version	Dependency Type
Microsoft.VisualStudio.Component.TypeScript.3.1	TypeScript 3.1 SDK	15.0.28218.60	Required
Microsoft.VisualStudio.Component.UWP.Support	Universal Windows Platform tools	15.9.28119.51	Required
Microsoft.VisualStudio.Component.VisualStudioData	Data sources and service references	15.6.27406.0	Required
Microsoft.VisualStudio.Component.Windows10SDK.17763	Windows 10 SDK (10.0.17763.0)	15.9.28307.102	Required
Microsoft.VisualStudio.ComponentGroup.UWP.Cordova	Universal Windows Platform tools for Cordova	15.9.28307.102	Required
Microsoft.VisualStudio.ComponentGroup.UWP.NetCoreAndStandard	.NET Native and .NET Standard	15.8.27906.1	Required
Microsoft.VisualStudio.ComponentGroup.UWP.Xamarin	Universal Windows Platform tools for Xamarin	15.9.28307.102	Required
Microsoft.VisualStudio.ComponentGroup.WebToolsExtensions	ASP.NET and web development	15.8.27825.0	Required
Microsoft.Component.VC.Runtime.OSSupport	Visual C++ runtime for UWP	15.6.27406.0	Optional
Microsoft.Net.Component.4.7.2.SDK	.NET Framework 4.7.2 SDK	15.8.27825.0	Optional
Microsoft.VisualStudio.Component.Graphics.Tools	Graphics debugger and GPU profiler for DirectX	15.6.27406.0	Optional
Microsoft.VisualStudio.Component.Graphics.Win81	Graphics Tools Windows 8.1 SDK	15.6.27406.0	Optional
Microsoft.VisualStudio.Component.Phone Emulator.15254	Windows 10 Mobile Emulator (Fall Creators Update)	15.0.27406.0	Optional
Microsoft.VisualStudio.Component.UWP.VC.ARM64	C++ Universal Windows Platform tools for ARM64	15.0.28125.51	Optional
Microsoft.VisualStudio.Component.VC.Corelde	Visual Studio C++ core features	15.6.27406.0	Optional
Microsoft.VisualStudio.Component.VC.Tools.ARM	Visual C++ compilers and libraries for ARM	15.8.27825.0	Optional
Microsoft.VisualStudio.Component.VC.Tools.ARM64	Visual C++ compilers and libraries for ARM64	15.9.28230.55	Optional

Component ID	Name	Version	Dependency Type
Microsoft.VisualStudio.Component.VC.Tools.x86.x64	VC++ 2017 version 15.9 v14.16 latest v141 tools	15.9.28230.55	Optional
Microsoft.VisualStudio.Component.Windows10SDK.10240	Windows 10 SDK (10.0.10240.0)	15.6.27406.0	Optional
Microsoft.VisualStudio.Component.Windows10SDK.10586	Windows 10 SDK (10.0.10586.0)	15.6.27406.0	Optional
Microsoft.VisualStudio.Component.Windows10SDK.14393	Windows 10 SDK (10.0.14393.0)	15.6.27406.0	Optional
Microsoft.VisualStudio.Component.Windows10SDK.15063/Desktop	Windows 10 SDK (10.0.15063.0) for Desktop C++ [x86 and x64]	15.6.27406.0	Optional
Microsoft.VisualStudio.Component.Windows10SDK.15063.UWP	Windows 10 SDK (10.0.15063.0) for UWP: C#, VB, JS	15.6.27406.0	Optional
Microsoft.VisualStudio.Component.Windows10SDK.15063.UWP.Native	Windows 10 SDK (10.0.15063.0) for UWP: C++	15.6.27406.0	Optional
Microsoft.VisualStudio.Component.Windows10SDK.16299/Desktop	Windows 10 SDK (10.0.16299.0) for Desktop C++ [x86 and x64]	15.6.27406.0	Optional
Microsoft.VisualStudio.Component.Windows10SDK.16299/Desktop.arm	Windows 10 SDK (10.0.16299.0) for Desktop C++ [ARM and ARM64]	15.6.27406.0	Optional
Microsoft.VisualStudio.Component.Windows10SDK.16299/UWP	Windows 10 SDK (10.0.16299.0) for UWP: C#, VB, JS	15.6.27406.0	Optional
Microsoft.VisualStudio.Component.Windows10SDK.16299.UWP.Native	Windows 10 SDK (10.0.16299.0) for UWP: C++	15.6.27406.0	Optional
Microsoft.VisualStudio.Component.Windows10SDK.17134	Windows 10 SDK (10.0.17134.0)	15.9.28307.102	Optional
Microsoft.VisualStudio.Component.Windows10SDK.IpOverUsb	USB Device Connectivity	15.9.28307.102	Optional
Microsoft.VisualStudio.ComponentGroup.UWP.VC	C++ Universal Windows Platform tools	15.9.28307.102	Optional

Component ID	Name	Version	Dependency Type
Microsoft.VisualStudio.ComponentGroup.Windows10SDK.15063	Windows 10 SDK (10.0.15063.0)	15.8.27825.0	Optional
Microsoft.VisualStudio.ComponentGroup.Windows10SDK.16299	Windows 10 SDK (10.0.16299.0)	15.8.27825.0	Optional

Visual Studio extension development

ID: Microsoft.VisualStudio.Workload.VisualStudioExtension

Description: Create add-ons and extensions for Visual Studio, including new commands, code analyzers and tool windows.

Components included by this workload

Component ID	Name	Version	Dependency Type
Microsoft.Component.ClickOnce	ClickOnce Publishing	15.8.27825.0	Required
Microsoft.Component.MSBuild	MSBuild	15.7.27520.0	Required
Microsoft.Net.Component.4.6.1.SDK	.NET Framework 4.6.1 SDK	15.6.27406.0	Required
Microsoft.Net.Component.4.6.1.TargetingPack	.NET Framework 4.6.1 targeting pack	15.6.27406.0	Required
Microsoft.Net.Component.4.6.TargetingPack	.NET Framework 4.6 targeting pack	15.6.27406.0	Required
Microsoft.Net.ComponentGroup.DevelopmentPrerequisites	.NET Framework 4.6.1 development tools	15.8.27825.0	Required
Microsoft.VisualStudio.Component.NuGet	NuGet package manager	15.9.28016.0	Required
Microsoft.VisualStudio.Component.PortableLibrary	.NET Portable Library targeting pack	15.6.27309.0	Required
Microsoft.VisualStudio.Component.Roslyn.Compiler	C# and Visual Basic Roslyn compilers	15.6.27309.0	Required
Microsoft.VisualStudio.Component.Roslyn.LanguageServices	C# and Visual Basic	15.8.27729.1	Required
Microsoft.VisualStudio.Component.Static.Analysis.Tools	Static analysis tools	15.0.26208.0	Required

Component ID	Name	Version	Dependency Type
Microsoft.VisualStudio.Component.VSSDK	Visual Studio SDK	15.8.27729.1	Required
Microsoft.VisualStudio.ComponentGroup.VisualStudioExtension.Prerequisites	Visual Studio extension development prerequisites	15.7.27625.0	Required
Microsoft.VisualStudio.Component.DiagnosticTools	.NET profiling tools	15.8.27729.1	Recommended
Microsoft.VisualStudio.Component.TextTemplating	Text Template Transformation	15.0.26208.0	Recommended
Component.Dotfuscator	PreEmptive Protection - Dotfuscator	15.0.26208.0	Optional
Microsoft.Component.CodeAnalysis.SDK	.NET Compiler Platform SDK	15.0.27729.1	Optional
Microsoft.Component.VC.Runtime.OSSupport	Visual C++ runtime for UWP	15.6.27406.0	Optional
Microsoft.VisualStudio.Component.AppInsights.Tools	Developer Analytics tools	15.8.27825.0	Optional
Microsoft.VisualStudio.Component.ClassDesigner	Class Designer	15.0.26208.0	Optional
Microsoft.VisualStudio.Component.DslTools	Modeling SDK	15.0.27005.2	Optional
Microsoft.VisualStudio.Component.VC.ATL	Visual C++ ATL for x86 and x64	15.7.27625.0	Optional
Microsoft.VisualStudio.Component.VC.ATLMFC	Visual C++ MFC for x86 and x64	15.7.27625.0	Optional
Microsoft.VisualStudio.Component.VC.Corelde	Visual Studio C++ core features	15.6.27406.0	Optional
Microsoft.VisualStudio.Component.VC.Tools.x86.x64	VC++ 2017 version 15.9 v14.16 latest v141 tools	15.9.28230.55	Optional

Mobile development with JavaScript

ID: Microsoft.VisualStudio.Workload.WebCrossPlat

Description: Build Android, iOS and UWP apps using Tools for Apache Cordova.

Components included by this workload

Component ID	Name	Version	Dependency Type
Component.CordovaToolset.6.3.1	Cordova 6.3.1 toolset	15.7.27625.0	Required
Component.WebSocket	WebSocket4Net	15.0.26606.0	Required
Microsoft.VisualStudio.Component.Cordova	Mobile development with JavaScript core features	15.0.26606.0	Required
Microsoft.VisualStudio.Component.JavaScript.Diagnostics	JavaScript diagnostics	15.8.27729.1	Required
Microsoft.VisualStudio.Component.JavaScript.ProjectSystem	JavaScript ProjectSystem and Shared Tooling	15.0.26606.0	Required
Microsoft.VisualStudio.Component.JavaScript.TypeScript	JavaScript and TypeScript language support	15.9.28125.51	Required
Microsoft.VisualStudio.Component.TypeScript.2.3	TypeScript 2.3 SDK	15.8.27729.1	Required
Microsoft.VisualStudio.Component.TypeScript.3.1	TypeScript 3.1 SDK	15.0.28218.60	Required
Microsoft.VisualStudio.ComponentGroup.WebToolsExtensions	ASP.NET and web development	15.8.27825.0	Required
Component.Android.SDK23.Private	Android SDK setup (API level 23) (local install for Mobile development with JavaScript / C++)	15.9.28016.0	Optional
Component.Google.Android.Emulator.API23.Private	Google Android Emulator (API Level 23) (local install)	15.6.27413.0	Optional
Component.HAXM.Private	Intel Hardware Accelerated Execution Manager (HAXM) (local install)	15.9.28307.421	Optional
Component.OpenJDK	Microsoft distribution OpenJDK	15.9.28125.51	Optional
Microsoft.Component.ClickOnce	ClickOnce Publishing	15.8.27825.0	Optional
Microsoft.Component.NetFX.Native	.NET Native	15.0.26208.0	Optional
Microsoft.VisualStudio.Component.AppInsights.Tools	Developer Analytics tools	15.8.27825.0	Optional
Microsoft.VisualStudio.Component.DiagnosticTools	.NET profiling tools	15.8.27729.1	Optional

Component ID	Name	Version	Dependency Type
Microsoft.VisualStudio.Component.Git	Git for Windows	15.0.26208.0	Optional
Microsoft.VisualStudio.Component.Graphics	Image and 3D model editors	15.6.27406.0	Optional
Microsoft.VisualStudio.Component.Phone Emulator.15254	Windows 10 Mobile Emulator (Fall Creators Update)	15.0.27406.0	Optional
Microsoft.VisualStudio.Component.SQL.CLR	CLR data types for SQL Server	15.0.26208.0	Optional
Microsoft.VisualStudio.Component.VisualStudioData	Data sources and service references	15.6.27406.0	Optional
Microsoft.VisualStudio.Component.Windows10SDK.17763	Windows 10 SDK (10.0.17763.0)	15.9.28307.102	Optional
Microsoft.VisualStudio.ComponentGroup.UWP.Cordova	Universal Windows Platform tools for Cordova	15.9.28307.102	Optional

Unaffiliated components

These are components that are not included with any workload, but may be selected as an individual component.

Component ID	Name	Version
Component.Android.Emulator	Visual Studio Emulator for Android	15.6.27413.0
Component.Android.NDK.R11C	Android NDK (R11C)	11.3.14
Component.Android.NDK.R11C_3264	Android NDK (R11C) (32bit)	11.3.16
Component.Android.SDK23	Android SDK setup (API level 23) (global install)	15.9.28107.0
Component.Android.SDK25	Android SDK setup (API level 25)	15.9.28107.0
Component.GitHub.VisualStudio	GitHub extension for Visual Studio	2.5.2.2500
Component.Google.Android.Emulator.API23.V2	Google Android Emulator (API Level 23) (global install)	15.6.27413.0
Component.Google.Android.Emulator.API25	Google Android Emulator (API Level 25)	15.7.27604.0
Microsoft.Component.Blend.SDK.WPF	Blend for Visual Studio SDK for .NET	15.6.27406.0
Microsoft.Component.HelpViewer	Help Viewer	15.9.28307.421

Component ID	Name	Version
Microsoft.VisualStudio.Component.DependencyValidation.Community	Dependency Validation	15.0.26208.0
Microsoft.VisualStudio.Component.GraphDocument	DGML editor	15.0.27005.2
Microsoft.VisualStudio.Component.LinqToSql	LINQ to SQL tools	15.6.27406.0
Microsoft.VisualStudio.Component.Phone.Emulator	Windows 10 Mobile Emulator (Anniversary Edition)	15.6.27406.0
Microsoft.VisualStudio.Component.Phone.Emulator.15063	Windows 10 Mobile Emulator (Creators Update)	15.6.27406.0
Microsoft.VisualStudio.Component.Runtime.Node.x86.6.4.0	Runtime for components based on Node.js v6.4.0 (x86)	15.7.27617.1
Microsoft.VisualStudio.Component.Runtime.Node.x86.7.4.0	Runtime for components based on Node.js v7.4.0 (x86)	15.7.27617.1
Microsoft.VisualStudio.Component.TypeScript.2.0	TypeScript 2.0 SDK	15.8.27729.1
Microsoft.VisualStudio.Component.TypeScript.2.1	TypeScript 2.1 SDK	15.8.27729.1
Microsoft.VisualStudio.Component.TypeScript.2.2	TypeScript 2.2 SDK	15.8.27729.1
Microsoft.VisualStudio.Component.TypeScript.2.5	TypeScript 2.5 SDK	15.6.27406.0
Microsoft.VisualStudio.Component.TypeScript.2.6	TypeScript 2.6 SDK	15.0.27729.1
Microsoft.VisualStudio.Component.TypeScript.2.7	TypeScript 2.7 SDK	15.0.27729.1
Microsoft.VisualStudio.Component.TypeScript.2.8	TypeScript 2.8 SDK	15.0.27729.1
Microsoft.VisualStudio.Component.TypeScript.2.9	TypeScript 2.9 SDK	15.0.27924.0
Microsoft.VisualStudio.Component.TypeScript.3.0	TypeScript 3.0 SDK	15.0.27924.0
Microsoft.VisualStudio.Component.VC.ATL.ARM	Visual C++ ATL for ARM	15.7.27625.0
Microsoft.VisualStudio.Component.VC.ATL.ARM.Spectre	Visual C++ ATL for ARM with Spectre Mitigations	15.7.27625.0

Component ID	Name	Version
Microsoft.VisualStudio.Component.VC.ATL.ARM64	Visual C++ ATL for ARM64	15.7.27625.0
Microsoft.VisualStudio.Component.VC.ATL.ARM64.Spectre	Visual C++ ATL for ARM64 with Spectre Mitigations	15.7.27625.0
Microsoft.VisualStudio.Component.VC.ATL.Spectre	Visual C++ ATL (x86/x64) with Spectre Mitigations	15.7.27625.0
Microsoft.VisualStudio.Component.VC.ATLMFC.Spectre	Visual C++ MFC for x86/x64 with Spectre Mitigations	15.7.27625.0
Microsoft.VisualStudio.Component.VC.ClangC2	Clang/C2 (experimental)	15.7.27520.0
Microsoft.VisualStudio.Component.VC.MFC.ARM	Visual C++ MFC for ARM	15.7.27625.0
Microsoft.VisualStudio.Component.VC.MFC.ARM.Spectre	Visual C++ MFC for ARM with Spectre Mitigations	15.7.27625.0
Microsoft.VisualStudio.Component.VC.MFC.ARM64	Visual C++ MFC for ARM64	15.7.27625.0
Microsoft.VisualStudio.Component.VC.MFC.ARM64.Spectre	Visual C++ MFC support for ARM64 with Spectre Mitigations	15.7.27625.0
Microsoft.VisualStudio.Component.VC.Runtimes.ARM.Spectre	VC++ 2017 version 15.9 v14.16 Libs for Spectre (ARM)	15.9.28230.55
Microsoft.VisualStudio.Component.VC.Runtimes.ARM64.Spectre	VC++ 2017 version 15.9 v14.16 Libs for Spectre (ARM64)	15.9.28230.55
Microsoft.VisualStudio.Component.VC.Runtimes.x86.x64.Spectre	VC++ 2017 version 15.9 v14.16 Libs for Spectre (x86 and x64)	15.9.28230.55
Microsoft.VisualStudio.Component.VC.Tools.14.11	VC++ 2017 version 15.4 v14.11 toolset	15.0.27924.0
Microsoft.VisualStudio.Component.VC.Tools.14.12	VC++ 2017 version 15.5 v14.12 toolset	15.0.27924.0
Microsoft.VisualStudio.Component.VC.Tools.14.13	VC++ 2017 version 15.6 v14.13 toolset	15.0.27924.0
Microsoft.VisualStudio.Component.VC.Tools.14.14	VC++ 2017 version 15.7 v14.14 toolset	15.0.27924.0
Microsoft.VisualStudio.Component.VC.Tools.14.15	VC++ 2017 version 15.8 v14.15 toolset	15.0.28230.55

When you set dependencies in your VSIX manifest, you must specify Component IDs only. Use the tables on this page to determine the minimum component dependencies. In some scenarios, this might mean that you specify only one component from a workload. In other scenarios, it might mean that you specify multiple components.

from a single workload or multiple components from multiple workloads. For more information, see the [How to: Migrate extensibility projects to Visual Studio](#) page.

For more information about how to use these IDs, see the [Use command-line parameters to install Visual Studio](#) page. And, for a list of workload and component IDs for other products, see the [Visual Studio workload and component IDs](#) page.

Visual Studio core editor (included with Visual Studio Community 2019)

ID: Microsoft.VisualStudio.Workload.CoreEditor

Description: The Visual Studio core shell experience, including syntax-aware code editing, source code control and work item management.

Components included by this workload

COMPONENT ID	NAME	VERSION	DEPENDENCY TYPE
Microsoft.VisualStudio.Component.CoreEditor	Visual Studio core editor	16.1.28811.260	Required
Microsoft.VisualStudio.Component.StartPageExperiment.Cpp	Visual Studio Start Page for C++ Users	16.0.28315.86	Optional

Azure development

ID: Microsoft.VisualStudio.Workload.Azure

Description: Azure SDKs, tools, and projects for developing cloud apps and creating resources using .NET Core and .NET Framework. Also includes tools for containerizing your application, including Docker support.

Components included by this workload

COMPONENT ID	NAME	VERSION	DEPENDENCY TYPE
Component.Microsoft.VisualStudio.RazorExtension	Razor Language Services	16.0.28714.129	Required
Component.Microsoft.VisualStudio.Web.AzureFunctions	Azure WebJobs Tools	16.0.28714.129	Required
Component.Microsoft.Web.LibraryManager	Library Manager	16.0.28315.86	Required
Microsoft.Component.MSBuild	MSBuild	16.4.29429.68	Required
Microsoft.Net.Component.4.5.2.TargetingPack	.NET Framework 4.5.2 targeting pack	16.0.28517.75	Required
Microsoft.Net.Component.4.5.TargetingPack	.NET Framework 4.5 targeting pack	16.0.28517.75	Required
Microsoft.Net.Component.4.6.1.TargetingPack	.NET Framework 4.6.1 targeting pack	16.0.28517.75	Required

Component ID	Name	Version	Dependency Type
Microsoft.Net.Component.4.7.2.TargetingPack	.NET Framework 4.7.2 targeting pack	16.0.28517.75	Required
Microsoft.Net.Component.4.8.SDK	.NET Framework 4.8 SDK	16.4.29313.120	Required
Microsoft.Net.ComponentGroup.DevelopmentPrerequisites	.NET Framework 4.7.2 development tools	16.3.29207.166	Required
Microsoft.NetCore.Component.DevelopmentTools	.NET Core development tools	16.4.29511.114	Required
Microsoft.NetCore.Component.SDK	.NET Core 3.1 SDK	16.4.29519.181	Required
Microsoft.NetCore.Component.Web	.NET Core development tools	16.4.29511.114	Required
Microsoft.VisualStudio.Component.Azure.AuthoringTools	Azure Authoring Tools	16.0.28625.61	Required
Microsoft.VisualStudio.Component.Azure.ClientLibs	Azure libraries for .NET	16.0.28315.86	Required
Microsoft.VisualStudio.Component.Azure.Compute.Emulator	Azure Compute Emulator	16.1.28810.153	Required
Microsoft.VisualStudio.Component.Azure.Storage.Emulator	Azure Storage Emulator	16.4.29313.120	Required
Microsoft.VisualStudio.Component.CloudExplorer	Cloud Explorer	16.0.28625.61	Required
Microsoft.VisualStudio.Component.Common.Azure.Tools	Connectivity and publishing tools	16.4.29409.204	Required
Microsoft.VisualStudio.Component.DockerTools	Container development tools	16.4.29409.204	Required
Microsoft.VisualStudio.Component.FSharp	F# language support	16.0.28315.86	Required
Microsoft.VisualStudio.Component.FSharp.WebTemplates	F# language support for web projects	16.3.29207.166	Required
Microsoft.VisualStudio.Component.IISExpress	IIS Express	16.0.28315.86	Required
Microsoft.VisualStudio.Component.IntelliCode	IntelliCode	0.1	Required

Component ID	Name	Version	Dependency Type
Microsoft.VisualStudio.Component.JavaScript.Diagnostics	JavaScript diagnostics	16.0.28517.75	Required
Microsoft.VisualStudio.Component.JavaScript.TypeScript	JavaScript and TypeScript language support	16.4.29409.204	Required
Microsoft.VisualStudio.Component.ManagedDesktop.Core	Managed Desktop Workload Core	16.4.29318.151	Required
Microsoft.VisualStudio.Component.MSODBC.SQL	SQL Server ODBC Driver	16.0.28625.61	Required
Microsoft.VisualStudio.Component.MSSQL.CMDLnUtils	SQL Server Command Line Utilities	16.0.28707.177	Required
Microsoft.VisualStudio.Component.NuGet	NuGet package manager	16.1.28829.92	Required
Microsoft.VisualStudio.Component.Roslyn.Compiler	C# and Visual Basic Roslyn compilers	16.0.28714.129	Required
Microsoft.VisualStudio.Component.Roslyn.LanguageServices	C# and Visual Basic	16.4.29429.68	Required
Microsoft.VisualStudio.Component.SQL.ADAL	SQL ADAL runtime	16.0.28517.75	Required
Microsoft.VisualStudio.Component.SQL.CLR	CLR data types for SQL Server	16.0.28315.86	Required
Microsoft.VisualStudio.Component.SQL.DataSources	Data sources for SQL Server support	16.0.28315.86	Required
Microsoft.VisualStudio.Component.SQL.LocalDB.Runtime	SQL Server Express 2016 LocalDB	16.0.28625.61	Required
Microsoft.VisualStudio.Component.SQL.SSDT	SQL Server Data Tools	16.3.29207.166	Required
Microsoft.VisualStudio.Component.TextTemplating	Text Template Transformation	16.0.28625.61	Required
Microsoft.VisualStudio.Component.TypeScript.3.7	TypeScript 3.7 SDK	16.0.29429.68	Required
Microsoft.VisualStudio.Component.Web	ASP.NET and web development tools	16.0.28517.75	Required
Microsoft.VisualStudio.ComponentGroup.Azure.Prerequisites	Azure development prerequisites	16.4.29409.204	Required

Component ID	Name	Version	Dependency Type
Microsoft.VisualStudio.ComponentGroup.AzureFunctions	Azure WebJobs Tools	16.0.28621.142	Required
Microsoft.VisualStudio.ComponentGroup.Web	ASP.NET and web development tools prerequisites	16.4.29318.151	Required
Microsoft.VisualStudio.ComponentGroup.WebToolsExtensions	ASP.NET and web development	16.0.28621.142	Required
Microsoft.Component.Azure.DataLake.Tools	Azure Data Lake and Stream Analytics Tools	16.4.29313.120	Recommended
Microsoft.Net.Component.4.5.1.TargetingPack	.NET Framework 4.5.1 targeting pack	16.0.28517.75	Recommended
Microsoft.Net.Component.4.6.TargetingPack	.NET Framework 4.6 targeting pack	16.0.28517.75	Recommended
Microsoft.Net.Component.4.TargetingPack	.NET Framework 4 targeting pack	16.0.28517.75	Recommended
Microsoft.Net.ComponentGroup.TargetingPacks.Common	.NET Framework 4 – 4.6 development tools	16.0.28516.191	Recommended
Microsoft.Net.Core.Component.SDK.2.1	.NET Core 2.1 LTS Runtime	16.4.29519.181	Recommended
Microsoft.VisualStudio.Component.AspNet45	Advanced ASP.NET features	16.0.28315.86	Recommended
Microsoft.VisualStudio.Component.Azure.Kubernetes.Tools	Visual Studio Tools for Kubernetes	16.0.28625.61	Recommended
Microsoft.VisualStudio.Component.Azure.PowerShell	Azure Powershell	16.4.29429.68	Recommended
Microsoft.VisualStudio.Component.Azure.ResourceManager.Tools	Azure Resource Manager core tools	16.4.29409.204	Recommended
Microsoft.VisualStudio.Component.Azure.ServiceFabric.Tools	Service Fabric Tools	16.4.29313.120	Recommended
Microsoft.VisualStudio.Component.Azure.Waverton	Azure Cloud Services core tools	16.4.29409.204	Recommended
Microsoft.VisualStudio.Component.Azure.Waverton.BuildTools	Azure Cloud Services build tools	16.3.29207.166	Recommended

Component ID	Name	Version	Dependency Type
Microsoft.VisualStudio.Component.DiagnosticTools	.NET profiling tools	16.4.29429.68	Recommended
Microsoft.VisualStudio.Component.WebDeploy	Web Deploy	16.0.28517.75	Recommended
Microsoft.VisualStudio.ComponentGroup.Azure.CloudServices	Azure Cloud Services tools	16.4.29409.204	Recommended
Microsoft.VisualStudio.ComponentGroup.Azure.ResourceManager.Tools	Azure Resource Manager tools	16.0.28528.71	Recommended
Microsoft.Net.Component.4.6.2.TargetingPack	.NET Framework 4.6.2 targeting pack	16.0.28517.75	Optional
Microsoft.Net.Component.4.7.1.TargetingPack	.NET Framework 4.7.1 targeting pack	16.0.28517.75	Optional
Microsoft.Net.Component.4.7.TargetingPack	.NET Framework 4.7 targeting pack	16.0.28517.75	Optional
Microsoft.Net.Component.4.8.TargetingPack	.NET Framework 4.8 targeting pack	16.4.29313.120	Optional
Microsoft.Net.ComponentGroup.4.6.1.DeveloperTools	.NET Framework 4.6.1 development tools	16.3.29207.166	Optional
Microsoft.Net.ComponentGroup.4.6.2.DeveloperTools	.NET Framework 4.6.2 development tools	16.3.29207.166	Optional
Microsoft.Net.ComponentGroup.4.7.1.DeveloperTools	.NET Framework 4.7.1 development tools	16.3.29207.166	Optional
Microsoft.Net.ComponentGroup.4.7.DeveloperTools	.NET Framework 4.7 development tools	16.3.29207.166	Optional
Microsoft.Net.ComponentGroup.4.8.DeveloperTools	.NET Framework 4.8 development tools	16.4.29318.151	Optional
Microsoft.VisualStudio.Component.Azure.Storage.AzCopy	Azure Storage AzCopy	16.0.28517.75	Optional
Microsoft.VisualStudio.Component.Wcf.Tooling	Windows Communication Foundation	16.0.28625.61	Optional

Data storage and processing

ID: Microsoft.VisualStudio.Workload.Data

Description: Connect, develop, and test data solutions with SQL Server, Azure Data Lake, or Hadoop.

Components included by this workload

Component ID	Name	Version	Dependency Type
Component.Microsoft.VisualStudio.RazorExtension	Razor Language Services	16.0.28714.129	Recommended
Component.Microsoft.Web.LibraryManager	Library Manager	16.0.28315.86	Recommended
Microsoft.Component.Azure.DataLake.Tools	Azure Data Lake and Stream Analytics Tools	16.4.29313.120	Recommended
Microsoft.Component_MSBuild	MSBuild	16.4.29429.68	Recommended
Microsoft.Net.Component.4.5.1.TargetingPack	.NET Framework 4.5.1 targeting pack	16.0.28517.75	Recommended
Microsoft.Net.Component.4.5.2.TargetingPack	.NET Framework 4.5.2 targeting pack	16.0.28517.75	Recommended
Microsoft.Net.Component.4.5.TargetingPack	.NET Framework 4.5 targeting pack	16.0.28517.75	Recommended
Microsoft.Net.Component.4.6.TargetingPack	.NET Framework 4.6 targeting pack	16.0.28517.75	Recommended
Microsoft.Net.Component.4.7.2.TargetingPack	.NET Framework 4.7.2 targeting pack	16.0.28517.75	Recommended
Microsoft.Net.Component.4.8.SDK	.NET Framework 4.8 SDK	16.4.29313.120	Recommended
Microsoft.Net.Component.4.TargetingPack	.NET Framework 4 targeting pack	16.0.28517.75	Recommended
Microsoft.Net.ComponentGroup.DevelopmentPrerequisites	.NET Framework 4.7.2 development tools	16.3.29207.166	Recommended
Microsoft.Net.ComponentGroup.TargetingPacks.Common	.NET Framework 4 – 4.6 development tools	16.0.28516.191	Recommended
Microsoft.NetCore.Component.SDK	.NET Core 3.1 SDK	16.4.29519.181	Recommended
Microsoft.VisualStudio.Component.Azure.AuthoringTools	Azure Authoring Tools	16.0.28625.61	Recommended
Microsoft.VisualStudio.Component.Azure.ClientLibs	Azure libraries for .NET	16.0.28315.86	Recommended
Microsoft.VisualStudio.Component.Azure.Compute.Emulator	Azure Compute Emulator	16.1.28810.153	Recommended

Component ID	Name	Version	Dependency Type
Microsoft.VisualStudio.Component.Azure.Storage Emulator	Azure Storage Emulator	16.4.29313.120	Recommended
Microsoft.VisualStudio.Component.Azure.Waverton	Azure Cloud Services core tools	16.4.29409.204	Recommended
Microsoft.VisualStudio.Component.Azure.Waverton.Build Tools	Azure Cloud Services build tools	16.3.29207.166	Recommended
Microsoft.VisualStudio.Component.CloudExplorer	Cloud Explorer	16.0.28625.61	Recommended
Microsoft.VisualStudio.Component.Common.Azure.Tools	Connectivity and publishing tools	16.4.29409.204	Recommended
Microsoft.VisualStudio.Component.DockerTools	Container development tools	16.4.29409.204	Recommended
Microsoft.VisualStudio.Component.IISExpress	IIS Express	16.0.28315.86	Recommended
Microsoft.VisualStudio.Component.JavaScript.Diagnostics	JavaScript diagnostics	16.0.28517.75	Recommended
Microsoft.VisualStudio.Component.JavaScript.TypeScript	JavaScript and TypeScript language support	16.4.29409.204	Recommended
Microsoft.VisualStudio.Component.ManagedDesktop.Core	Managed Desktop Workload Core	16.4.29318.151	Recommended
Microsoft.VisualStudio.Component.MSODBC.SQL	SQL Server ODBC Driver	16.0.28625.61	Recommended
Microsoft.VisualStudio.Component.MSSQL.CMDLnUtils	SQL Server Command Line Utilities	16.0.28707.177	Recommended
Microsoft.VisualStudio.Component.NuGet	NuGet package manager	16.1.28829.92	Recommended
Microsoft.VisualStudio.Component.Roslyn.Compiler	C# and Visual Basic Roslyn compilers	16.0.28714.129	Recommended
Microsoft.VisualStudio.Component.Roslyn.LanguageServices	C# and Visual Basic	16.4.29429.68	Recommended
Microsoft.VisualStudio.Component.SQL.ADAL	SQL ADAL runtime	16.0.28517.75	Recommended
Microsoft.VisualStudio.Component.SQL.CLR	CLR data types for SQL Server	16.0.28315.86	Recommended

Component ID	Name	Version	Dependency Type
Microsoft.VisualStudio.Component.SQL.DataSources	Data sources for SQL Server support	16.0.28315.86	Recommended
Microsoft.VisualStudio.Component.SQL.LocalDB.Runtime	SQL Server Express 2016 LocalDB	16.0.28625.61	Recommended
Microsoft.VisualStudio.Component.SQL.SSDT	SQL Server Data Tools	16.3.29207.166	Recommended
Microsoft.VisualStudio.Component.TextTemplating	Text Template Transformation	16.0.28625.61	Recommended
Microsoft.VisualStudio.Component.TypeScript.3.7	TypeScript 3.7 SDK	16.0.29429.68	Recommended
Microsoft.VisualStudio.Component.Web	ASP.NET and web development tools	16.0.28517.75	Recommended
Microsoft.VisualStudio.ComponentGroup.Web	ASP.NET and web development tools prerequisites	16.4.29318.151	Recommended
Microsoft.VisualStudio.ComponentGroup.WebToolsExtensions	ASP.NET and web development	16.0.28621.142	Recommended
Microsoft.Net.Component.4.6.1.TargetingPack	.NET Framework 4.6.1 targeting pack	16.0.28517.75	Optional
Microsoft.VisualStudio.Component.FSharp.Desktop	F# desktop language support	16.0.28315.86	Optional

Data science and analytical applications

ID: Microsoft.VisualStudio.Workload.DataScience

Description: Languages and tooling for creating data science applications, including Python and F#.

Components included by this workload

Component ID	Name	Version	Dependency Type
Microsoft.Component.PythonTools	Python language support	16.4.29429.68	Recommended
Microsoft.Component.PythonTools.Minicondax64	Python miniconda	16.2.29003.222	Recommended
Microsoft.Component.PythonTools.Web	Python web support	16.0.28517.75	Recommended
Microsoft.Net.Component.4.6.1.TargetingPack	.NET Framework 4.6.1 targeting pack	16.0.28517.75	Recommended

Component ID	Name	Version	Dependency Type
Microsoft.VisualStudio.Component.Common.Azure.Tools	Connectivity and publishing tools	16.4.29409.204	Recommended
Microsoft.VisualStudio.Component.FSharp.Desktop	F# desktop language support	16.0.28315.86	Recommended
Microsoft.VisualStudio.Component.JavaScript.TypeScript	JavaScript and TypeScript language support	16.4.29409.204	Recommended
Microsoft.VisualStudio.Component.NuGet	NuGet package manager	16.1.28829.92	Recommended
Microsoft.VisualStudio.Component.Roslyn.Compiler	C# and Visual Basic Roslyn compilers	16.0.28714.129	Recommended
Microsoft.VisualStudio.Component.Roslyn.LanguageServices	C# and Visual Basic	16.4.29429.68	Recommended
Microsoft.VisualStudio.Component.TypeScript.3.7	TypeScript 3.7 SDK	16.0.29429.68	Recommended
Microsoft.VisualStudio.Component.WebDeploy	Web Deploy	16.0.28517.75	Recommended
Microsoft.VisualStudio.ComponentGroup.WebToolsExtensions	ASP.NET and web development	16.0.28621.142	Recommended
Microsoft.ComponentGroup.PythonTools.NativeDevelopment	Python native development tools	16.2.29020.229	Optional
Microsoft.VisualStudio.Component.Graphics.Tools	Graphics debugger and GPU profiler for DirectX	16.0.28625.61	Optional
Microsoft.VisualStudio.Component.VC.Corelde	C++ core features	16.0.28625.61	Optional
Microsoft.VisualStudio.Component.VC.DiagnosticTools	C++ profiling tools	16.4.29429.68	Optional
Microsoft.VisualStudio.Component.VC.Tools.x86.x64	MSVC v142 - VS 2019 C++ x64/x86 build tools (v14.24)	16.4.29409.204	Optional
Microsoft.VisualStudio.Component.Windows10SDK	Windows Universal C Runtime	16.4.29409.204	Optional
Microsoft.VisualStudio.Component.Windows10SDK.18362.2	Windows 10 SDK (10.0.18362.0)	16.1.28829.92	Optional

.NET desktop development

ID: Microsoft.VisualStudio.Workload.ManagedDesktop

Description: Build WPF, Windows Forms, and console applications using C#, Visual Basic, and F# with .NET Core and .NET Framework.

Components included by this workload

COMPONENT ID	NAME	VERSION	DEPENDENCY TYPE
Microsoft.Component.MSBuild	MSBuild	16.4.29429.68	Required
Microsoft.Net.Component.4.7.2.TargetingPack	.NET Framework 4.7.2 targeting pack	16.0.28517.75	Required
Microsoft.Net.Component.4.8.SDK	.NET Framework 4.8 SDK	16.4.29313.120	Required
Microsoft.Net.ComponentGroup.DevelopmentPrerequisites	.NET Framework 4.7.2 development tools	16.3.29207.166	Required
Microsoft.NetCore.Component.SDK	.NET Core 3.1 SDK	16.4.29519.181	Required
Microsoft.VisualStudio.Component.IntelliCode	IntelliCode	0.1	Required
Microsoft.VisualStudio.Component.ManagedDesktop.Core	Managed Desktop Workload Core	16.4.29318.151	Required
Microsoft.VisualStudio.Component.ManagedDesktop.Prerequisites	.NET desktop development tools	16.4.29429.68	Required
Microsoft.VisualStudio.Component.Roslyn.Compiler	C# and Visual Basic Roslyn compilers	16.0.28714.129	Required
Microsoft.VisualStudio.Component.Roslyn.LanguageServices	C# and Visual Basic	16.4.29429.68	Required
Microsoft.VisualStudio.Component.SQL.CLR	CLR data types for SQL Server	16.0.28315.86	Required
Microsoft.VisualStudio.Component.TextTemplating	Text Template Transformation	16.0.28625.61	Required
Component.Microsoft.VisualStudio.LiveShare	Live Share	1.0.1076	Recommended
Microsoft.ComponentGroup.Blend	Blend for Visual Studio	16.0.28315.86	Recommended
Microsoft.Net.Component.4.5.1.TargetingPack	.NET Framework 4.5.1 targeting pack	16.0.28517.75	Recommended

Component ID	Name	Version	Dependency Type
Microsoft.Net.Component.4.5.2.TargetingPack	.NET Framework 4.5.2 targeting pack	16.0.28517.75	Recommended
Microsoft.Net.Component.4.5.TargetingPack	.NET Framework 4.5 targeting pack	16.0.28517.75	Recommended
Microsoft.Net.Component.4.6.1.TargetingPack	.NET Framework 4.6.1 targeting pack	16.0.28517.75	Recommended
Microsoft.Net.Component.4.6.TargetingPack	.NET Framework 4.6 targeting pack	16.0.28517.75	Recommended
Microsoft.Net.Component.4.TargetingPack	.NET Framework 4 targeting pack	16.0.28517.75	Recommended
Microsoft.Net.ComponentGroup.TargetingPacks.Common	.NET Framework 4 – 4.6 development tools	16.0.28516.191	Recommended
Microsoft.Net.Core.Component.SDK.2.1	.NET Core 2.1 LTS Runtime	16.4.29519.181	Recommended
Microsoft.NetCore.Component.DevelopmentTools	.NET Core development tools	16.4.29511.114	Recommended
Microsoft.VisualStudio.Component.Debugger.JustInTime	Just-In-Time debugger	16.0.28517.75	Recommended
Microsoft.VisualStudio.Component.DiagnosticTools	.NET profiling tools	16.4.29429.68	Recommended
Microsoft.VisualStudio.Component.EntityFramework	Entity Framework 6 tools	16.0.28315.86	Recommended
Microsoft.VisualStudio.Component.FSharp	F# language support	16.0.28315.86	Recommended
Microsoft.VisualStudio.Component.NuGet	NuGet package manager	16.1.28829.92	Recommended
Component.Dotfuscator	PreEmptive Protection - Dotfuscator	16.0.28528.71	Optional
Component.Microsoft.VisualStudio.RazorExtension	Razor Language Services	16.0.28714.129	Optional
Component.Microsoft.Web.LibraryManager	Library Manager	16.0.28315.86	Optional
Microsoft.Net.Component.4.6.2.TargetingPack	.NET Framework 4.6.2 targeting pack	16.0.28517.75	Optional
Microsoft.Net.Component.4.7.1.TargetingPack	.NET Framework 4.7.1 targeting pack	16.0.28517.75	Optional

Component ID	Name	Version	Dependency Type
Microsoft.Net.Component.4.7.TargetingPack	.NET Framework 4.7 targeting pack	16.0.28517.75	Optional
Microsoft.Net.Component.4.8.TargetingPack	.NET Framework 4.8 targeting pack	16.4.29313.120	Optional
Microsoft.Net.ComponentGroup.4.6.1.DeveloperTools	.NET Framework 4.6.1 development tools	16.3.29207.166	Optional
Microsoft.Net.ComponentGroup.4.6.2.DeveloperTools	.NET Framework 4.6.2 development tools	16.3.29207.166	Optional
Microsoft.Net.ComponentGroup.4.7.1.DeveloperTools	.NET Framework 4.7.1 development tools	16.3.29207.166	Optional
Microsoft.Net.ComponentGroup.4.7.DeveloperTools	.NET Framework 4.7 development tools	16.3.29207.166	Optional
Microsoft.Net.ComponentGroup.4.8.DeveloperTools	.NET Framework 4.8 development tools	16.4.29318.151	Optional
Microsoft.VisualStudio.Component.Common.Azure.Tools	Connectivity and publishing tools	16.4.29409.204	Optional
Microsoft.VisualStudio.Component.DockerTools	Container development tools	16.4.29409.204	Optional
Microsoft.VisualStudio.Component.FSharp.Desktop	F# desktop language support	16.0.28315.86	Optional
Microsoft.VisualStudio.Component.IISExpress	IIS Express	16.0.28315.86	Optional
Microsoft.VisualStudio.Component.JavaScript.Diagnostics	JavaScript diagnostics	16.0.28517.75	Optional
Microsoft.VisualStudio.Component.JavaScript.TypeScript	JavaScript and TypeScript language support	16.4.29409.204	Optional
Microsoft.VisualStudio.Component.MSODBC.SQL	SQL Server ODBC Driver	16.0.28625.61	Optional
Microsoft.VisualStudio.Component.MSSQL.CMDLnUtils	SQL Server Command Line Utilities	16.0.28707.177	Optional
Microsoft.VisualStudio.Component.PortableLibrary	.NET Portable Library targeting pack	16.0.28517.75	Optional
Microsoft.VisualStudio.Component.SQL.ADAL	SQL ADAL runtime	16.0.28517.75	Optional
Microsoft.VisualStudio.Component.SQL.DataSources	Data sources for SQL Server support	16.0.28315.86	Optional

Component ID	Name	Version	Dependency Type
Microsoft.VisualStudio.Component.SQL.LocalDB.Runtime	SQL Server Express 2016 LocalDB	16.0.28625.61	Optional
Microsoft.VisualStudio.Component.SQL.SSDT	SQL Server Data Tools	16.3.29207.166	Optional
Microsoft.VisualStudio.Component.TypeScript.3.7	TypeScript 3.7 SDK	16.0.29429.68	Optional
Microsoft.VisualStudio.Component.Wcf.Tooling	Windows Communication Foundation	16.0.28625.61	Optional
Microsoft.VisualStudio.Component.Web	ASP.NET and web development tools	16.0.28517.75	Optional
Microsoft.VisualStudio.Component.Windows10SDK.18362	Windows 10 SDK (10.0.18362.0)	16.1.28829.92	Optional
Microsoft.VisualStudio.ComponentGroup.MSIX.Packaging	MSIX Packaging Tools	16.4.29409.204	Optional
Microsoft.VisualStudio.ComponentGroup.Web	ASP.NET and web development tools prerequisites	16.4.29318.151	Optional
Microsoft.VisualStudio.ComponentGroup.WebToolsExtensions	ASP.NET and web development	16.0.28621.142	Optional

Game development with Unity

ID: Microsoft.VisualStudio.Workload.ManagedGame

Description: Create 2D and 3D games with Unity, a powerful cross-platform development environment.

Components included by this workload

Component ID	Name	Version	Dependency Type
Microsoft.Net.Component.3.5.DeveloperTools	.NET Framework 3.5 development tools	16.0.28517.75	Required
Microsoft.Net.Component.4.7.1.TargetingPack	.NET Framework 4.7.1 targeting pack	16.0.28517.75	Required
Microsoft.VisualStudio.Component.NuGet	NuGet package manager	16.1.28829.92	Required
Microsoft.VisualStudio.Component.Roslyn.Compiler	C# and Visual Basic Roslyn compilers	16.0.28714.129	Required

COMPONENT ID	NAME	VERSION	DEPENDENCY TYPE
Microsoft.VisualStudio.Component.Roslyn.LanguageServices	C# and Visual Basic	16.4.29429.68	Required
Microsoft.VisualStudio.Component.Unity	Visual Studio Tools for Unity	16.0.28315.86	Required
Component.UnityEngine.x64	Unity 2019.2 64-bit Editor	16.4.29429.68	Recommended
Component.UnityEngine.x86	Unity 5.6 32-bit Editor	16.1.28811.260	Recommended

Linux development with C++

ID: Microsoft.VisualStudio.Workload.NativeCrossPlat

Description: Create and debug applications running in a Linux environment.

Components included by this workload

COMPONENT ID	NAME	VERSION	DEPENDENCY TYPE
Component.MDD.Linux	C++ for Linux Development	16.4.29511.114	Required
Microsoft.VisualStudio.Component.IntelliCode	IntelliCode	0.1	Required
Microsoft.VisualStudio.Component.VC.CoreIDE	C++ core features	16.0.28625.61	Required
Component.Linux.CMake	C++ CMake tools for Linux	16.2.29003.222	Recommended
Microsoft.VisualStudio.ComponentGroup.WebToolsExtensions	ASP.NET and web development	16.0.28621.142	Recommended
Component.MDD.Linux.GCC.arm	Embedded and IoT development tools	16.4.29429.68	Optional

Desktop development with C++

ID: Microsoft.VisualStudio.Workload.NativeDesktop

Description: Build modern C++ apps for Windows using tools of your choice, including MSVC, Clang, CMake, or MSBuild.

Components included by this workload

COMPONENT ID	NAME	VERSION	DEPENDENCY TYPE
Microsoft.Component.MSBuild	MSBuild	16.4.29429.68	Required

Component ID	Name	Version	Dependency Type
Microsoft.VisualStudio.Component.IntelliCode	IntelliCode	0.1	Required
Microsoft.VisualStudio.Component.Roslyn.Compiler	C# and Visual Basic Roslyn compilers	16.0.28714.129	Required
Microsoft.VisualStudio.Component.TextTemplating	Text Template Transformation	16.0.28625.61	Required
Microsoft.VisualStudio.Component.VC.CoreLDE	C++ core features	16.0.28625.61	Required
Microsoft.VisualStudio.Component.VC.Redist.14.Latest	C++ 2019 Redistributable Update	16.4.29429.68	Required
Microsoft.VisualStudio.ComponentGroup.NativeDesktop.Core	C++ core desktop features	16.2.29012.281	Required
Component.Microsoft.VisualStudio.LiveShare	Live Share	1.0.1076	Recommended
Microsoft.VisualStudio.Component.Debugger.JustInTime	Just-In-Time debugger	16.0.28517.75	Recommended
Microsoft.VisualStudio.Component.Graphics.Tools	Graphics debugger and GPU profiler for DirectX	16.0.28625.61	Recommended
Microsoft.VisualStudio.Component.NuGet	NuGet package manager	16.1.28829.92	Recommended
Microsoft.VisualStudio.Component.VC.ASAN	C++ AddressSanitizer (Experimental)	16.4.29429.68	Recommended
Microsoft.VisualStudio.Component.VC.ATL	C++ ATL for latest v142 build tools (x86 & x64)	16.4.29313.120	Recommended
Microsoft.VisualStudio.Component.VC.CMake.Project	C++ CMake tools for Windows	16.3.29103.31	Recommended
Microsoft.VisualStudio.Component.VC.DiagnosticTools	C++ profiling tools	16.4.29429.68	Recommended
Microsoft.VisualStudio.Component.VC.TestAdapterForBoostTest	Test Adapter for Boost.Test	16.0.28517.75	Recommended
Microsoft.VisualStudio.Component.VC.TestAdapterForGoogleTest	Test Adapter for Google Test	16.0.28517.75	Recommended
Microsoft.VisualStudio.Component.VC.Tools.x86.x64	MSVC v142 - VS 2019 C++ x64/x86 build tools (v14.24)	16.4.29409.204	Recommended

Component ID	Name	Version	Dependency Type
Microsoft.VisualStudio.Component.Windows10SDK.18362	Windows 10 SDK (10.0.18362.0)	16.1.28829.92	Recommended
Microsoft.VisualStudio.ComponentGroup.WebToolsExtensions.CMake	JSON editor	16.3.29207.166	Recommended
Component.Incredibuild	Incredibuild - Build Acceleration	16.0.28528.71	Optional
Component.IncredibuildMenu	IncredibuildMenu	1.5.0.10	Optional
Microsoft.Component.VC.Runtime.UCRTSDK	Windows Universal CRT SDK	16.0.28625.61	Optional
Microsoft.Net.Component.4.6.1.TargetingPack	.NET Framework 4.6.1 targeting pack	16.0.28517.75	Optional
Microsoft.Net.Component.4.8.SDK	.NET Framework 4.8 SDK	16.4.29313.120	Optional
Microsoft.VisualStudio.Component.JavaScript.Diagnostics	JavaScript diagnostics	16.0.28517.75	Optional
Microsoft.VisualStudio.Component.VC.140	MSVC v140 - VS 2015 C++ build tools (v14.00)	16.0.28625.61	Optional
Microsoft.VisualStudio.Component.VC.ATLMFC	C++ MFC for latest v142 build tools (x86 & x64)	16.4.29313.120	Optional
Microsoft.VisualStudio.Component.VC.CLI.Support	C++/CLI support for v142 build tools (14.24)	16.4.29409.204	Optional
Microsoft.VisualStudio.Component.VC.Llvm.Clang	C++ Clang Compiler for Windows (9.0.0)	16.4.29511.114	Optional
Microsoft.VisualStudio.Component.VC.Llvm.ClangToolset	C++ Clang-cl for v142 build tools (x64/x86)	16.3.29207.166	Optional
Microsoft.VisualStudio.Component.VC.Modules.x86.x64	C++ Modules for v142 build tools (x64/x86 – experimental)	16.0.28625.61	Optional
Microsoft.VisualStudio.Component.VC.v141.x86.x64	MSVC v141 - VS 2017 C++ x64/x86 build tools (v14.16)	16.1.28829.92	Optional
Microsoft.VisualStudio.Component.Windows10SDK.16299	Windows 10 SDK (10.0.16299.0)	16.0.28517.75	Optional

Component ID	Name	Version	Dependency Type
Microsoft.VisualStudio.Component.Windows10SDK.17134	Windows 10 SDK (10.0.17134.0)	16.0.28517.75	Optional
Microsoft.VisualStudio.Component.Windows10SDK.17763	Windows 10 SDK (10.0.17763.0)	16.0.28517.75	Optional
Microsoft.VisualStudio.ComponentGroup.NativeDesktop.Llvm.Clang	C++ Clang tools for Windows (9.0.0 - x64/x86)	16.4.29511.114	Optional

Game development with C++

ID: Microsoft.VisualStudio.Workload.NativeGame

Description: Use the full power of C++ to build professional games powered by DirectX, Unreal, or Cocos2d.

Components included by this workload

Component ID	Name	Version	Dependency Type
Microsoft.VisualStudio.Component.IntelliCode	IntelliCode	0.1	Required
Microsoft.VisualStudio.Component.VC.CoreRedist	C++ core features	16.0.28625.61	Required
Microsoft.VisualStudio.Component.VC.Redist.14.Latest	C++ 2019 Redistributable Update	16.4.29429.68	Required
Microsoft.VisualStudio.Component.VC.Tools.x86.x64	MSVC v142 - VS 2019 C++ x64/x86 build tools (v14.24)	16.4.29409.204	Required
Microsoft.VisualStudio.Component.Windows10SDK	Windows Universal C Runtime	16.4.29409.204	Required
Microsoft.VisualStudio.Component.Graphics.Tools	Graphics debugger and GPU profiler for DirectX	16.0.28625.61	Recommended
Microsoft.VisualStudio.Component.VC.ASAN	C++ AddressSanitizer (Experimental)	16.4.29429.68	Recommended
Microsoft.VisualStudio.Component.VC.DiagnosticTools	C++ profiling tools	16.4.29429.68	Recommended
Microsoft.VisualStudio.Component.Windows10SDK.18362	Windows 10 SDK (10.0.18362.0)	16.1.28829.92	Recommended
Component.Android.NDK.R16B	Android NDK (R16B)	16.4.29519.181	Optional

Component ID	Name	Version	Dependency Type
Component.Android.SDK25.Private	Android SDK setup (API level 25) (local install for Mobile development with C++)	16.0.28625.61	Optional
Component.Ant	Apache Ant (1.9.3)	1.9.3.8	Optional
Component.Cocos	Cocos	16.0.28315.86	Optional
Component.Incredibuild	Incredibuild - Build Acceleration	16.0.28528.71	Optional
Component.IncredibuildMenu	IncredibuildMenu	1.5.0.10	Optional
Component.MDD.Android	C++ Android development tools	16.0.28517.75	Optional
Component.OpenJDK	OpenJDK (Microsoft distribution)	16.1.28811.260	Optional
Component.Unreal	Unreal Engine installer	16.1.28810.153	Optional
Component.Unreal.Android	Android IDE support for Unreal engine	16.1.28810.153	Optional
Microsoft.Net.Component.4.5.1.TargetingPack	.NET Framework 4.5.1 targeting pack	16.0.28517.75	Optional
Microsoft.Net.Component.4.5.2.TargetingPack	.NET Framework 4.5.2 targeting pack	16.0.28517.75	Optional
Microsoft.Net.Component.4.5.TargetingPack	.NET Framework 4.5 targeting pack	16.0.28517.75	Optional
Microsoft.Net.Component.4.6.2.TargetingPack	.NET Framework 4.6.2 targeting pack	16.0.28517.75	Optional
Microsoft.Net.Component.4.6.TargetingPack	.NET Framework 4.6 targeting pack	16.0.28517.75	Optional
Microsoft.Net.Component.4.7.2.TargetingPack	.NET Framework 4.7.2 targeting pack	16.0.28517.75	Optional
Microsoft.Net.Component.4.8.SDK	.NET Framework 4.8 SDK	16.4.29313.120	Optional
Microsoft.Net.Component.4.TargetingPack	.NET Framework 4 targeting pack	16.0.28517.75	Optional
Microsoft.Net.ComponentGroup.DevelopmentPrerequisites	.NET Framework 4.7.2 development tools	16.3.29207.166	Optional

Component ID	Name	Version	Dependency Type
Microsoft.Net.ComponentGroup.TargetingPacks.Common	.NET Framework 4 – 4.6 development tools	16.0.28516.191	Optional
Microsoft.VisualStudio.Component.NuGet.BuildTools	NuGet targets and build tasks	16.1.28829.92	Optional
Microsoft.VisualStudio.Component.Roslyn.Compiler	C# and Visual Basic Roslyn compilers	16.0.28714.129	Optional
Microsoft.VisualStudio.Component.Roslyn.LanguageServices	C# and Visual Basic	16.4.29429.68	Optional
Microsoft.VisualStudio.Component.Windows10SDK.16299	Windows 10 SDK (10.0.16299.0)	16.0.28517.75	Optional
Microsoft.VisualStudio.Component.Windows10SDK.17134	Windows 10 SDK (10.0.17134.0)	16.0.28517.75	Optional
Microsoft.VisualStudio.Component.Windows10SDK.17763	Windows 10 SDK (10.0.17763.0)	16.0.28517.75	Optional

Mobile development with C++

ID: Microsoft.VisualStudio.Workload.NativeMobile

Description: Build cross-platform applications for iOS, Android or Windows using C++.

Components included by this workload

Component ID	Name	Version	Dependency Type
Component.Android.SDK25.Private	Android SDK setup (API level 25) (local install for Mobile development with C++)	16.0.28625.61	Required
Component.OpenJDK	OpenJDK (Microsoft distribution)	16.1.28811.260	Required
Microsoft.VisualStudio.Component.IntelliCode	IntelliCode	0.1	Required
Microsoft.VisualStudio.Component.VC.Corelde	C++ core features	16.0.28625.61	Required
Component.Android.NDK.R16B	Android NDK (R16B)	16.4.29519.181	Recommended
Component.Ant	Apache Ant (1.9.3)	1.9.3.8	Recommended

COMPONENT ID	NAME	VERSION	DEPENDENCY TYPE
Component.MDD.Android	C++ Android development tools	16.0.28517.75	Recommended
Component.Android.NDK.R16B_3264	Android NDK (R16B) (32bit)	16.4.29519.181	Optional
Component.Google.Android.Emulator.API25.Private	Google Android Emulator (API Level 25) (local install)	16.1.28810.153	Optional
Component.HAXM.Private	Intel Hardware Accelerated Execution Manager (HAXM) (local install)	16.0.28528.71	Optional
Component.Incredibuild	Incredibuild - Build Acceleration	16.0.28528.71	Optional
Component.IncredibuildMenu	IncredibuildMenu	1.5.0.10	Optional
Component.MDD.iOS	C++ iOS development tools	16.0.28517.75	Optional

.NET Core cross-platform development

ID: Microsoft.VisualStudio.Workload.NetCoreTools

Description: Build cross-platform applications using .NET Core, ASP.NET Core, HTML/JavaScript, and Containers including Docker support.

Components included by this workload

COMPONENT ID	NAME	VERSION	DEPENDENCY TYPE
Component.Microsoft.VisualStudio.RazorExtension	Razor Language Services	16.0.28714.129	Required
Component.Microsoft.Web.LibraryManager	Library Manager	16.0.28315.86	Required
Microsoft.Component_MSBUILD	MSBuild	16.4.29429.68	Required
Microsoft.Net.Component.4.5.2.TargetingPack	.NET Framework 4.5.2 targeting pack	16.0.28517.75	Required
Microsoft.Net.Component.4.5.TargetingPack	.NET Framework 4.5 targeting pack	16.0.28517.75	Required
Microsoft.Net.Component.4.6.1.TargetingPack	.NET Framework 4.6.1 targeting pack	16.0.28517.75	Required
Microsoft.Net.Component.4.7.2.TargetingPack	.NET Framework 4.7.2 targeting pack	16.0.28517.75	Required

Component ID	Name	Version	Dependency Type
Microsoft.Net.Component.4.8.SDK	.NET Framework 4.8 SDK	16.4.29313.120	Required
Microsoft.Net.ComponentGroup.DevelopmentPrerequisites	.NET Framework 4.7.2 development tools	16.3.29207.166	Required
Microsoft.NetCore.Component.DevelopmentTools	.NET Core development tools	16.4.29511.114	Required
Microsoft.NetCore.Component.SDK	.NET Core 3.1 SDK	16.4.29519.181	Required
Microsoft.NetCore.Component.Web	.NET Core development tools	16.4.29511.114	Required
Microsoft.VisualStudio.Component.Common.Azure.Tools	Connectivity and publishing tools	16.4.29409.204	Required
Microsoft.VisualStudio.Component.DockerTools	Container development tools	16.4.29409.204	Required
Microsoft.VisualStudio.Component.FSharp	F# language support	16.0.28315.86	Required
Microsoft.VisualStudio.Component.FSharp.WebTemplates	F# language support for web projects	16.3.29207.166	Required
Microsoft.VisualStudio.Component.IISExpress	IIS Express	16.0.28315.86	Required
Microsoft.VisualStudio.Component.IntelliCode	IntelliCode	0.1	Required
Microsoft.VisualStudio.Component.JavaScript.Diagnostics	JavaScript diagnostics	16.0.28517.75	Required
Microsoft.VisualStudio.Component.JavaScript.TypeScript	JavaScript and TypeScript language support	16.4.29409.204	Required
Microsoft.VisualStudio.Component.ManagedDesktop.Core	Managed Desktop Workload Core	16.4.29318.151	Required
Microsoft.VisualStudio.Component.MSODBC.SQL	SQL Server ODBC Driver	16.0.28625.61	Required
Microsoft.VisualStudio.Component.MSSQL.CMDLnUtils	SQL Server Command Line Utilities	16.0.28707.177	Required
Microsoft.VisualStudio.Component.NuGet	NuGet package manager	16.1.28829.92	Required

Component ID	Name	Version	Dependency Type
Microsoft.VisualStudio.Component.Roslyn.Compiler	C# and Visual Basic Roslyn compilers	16.0.28714.129	Required
Microsoft.VisualStudio.Component.Roslyn.LanguageServices	C# and Visual Basic	16.4.29429.68	Required
Microsoft.VisualStudio.Component.SQL.ADAL	SQL ADAL runtime	16.0.28517.75	Required
Microsoft.VisualStudio.Component.SQL.CLR	CLR data types for SQL Server	16.0.28315.86	Required
Microsoft.VisualStudio.Component.SQL.DataSources	Data sources for SQL Server support	16.0.28315.86	Required
Microsoft.VisualStudio.Component.SQL.LocalDB.Runtime	SQL Server Express 2016 LocalDB	16.0.28625.61	Required
Microsoft.VisualStudio.Component.SQL.SSDT	SQL Server Data Tools	16.3.29207.166	Required
Microsoft.VisualStudio.Component.TextTemplating	Text Template Transformation	16.0.28625.61	Required
Microsoft.VisualStudio.Component.TypeScript.3.7	TypeScript 3.7 SDK	16.0.29429.68	Required
Microsoft.VisualStudio.ComponentGroup.Web	ASP.NET and web development tools prerequisites	16.4.29318.151	Required
Microsoft.VisualStudio.ComponentGroup.WebToolsExtensions	ASP.NET and web development	16.0.28621.142	Required
Component.Microsoft.VisualStudio.LiveShare	Live Share	1.0.1076	Recommended
Component.Microsoft.VisualStudio.Web.AzureFunctions	Azure WebJobs Tools	16.0.28714.129	Recommended
Microsoft.Net.Core.Component.SDK.2.1	.NET Core 2.1 LTS Runtime	16.4.29519.181	Recommended
Microsoft.VisualStudio.Component.AppInsights.Tools	Developer Analytics tools	16.4.29429.68	Recommended
Microsoft.VisualStudio.Component.Azure.AuthoringTools	Azure Authoring Tools	16.0.28625.61	Recommended
Microsoft.VisualStudio.Component.Azure.ClientLibs	Azure libraries for .NET	16.0.28315.86	Recommended

COMPONENT ID	NAME	VERSION	DEPENDENCY TYPE
Microsoft.VisualStudio.Component.Azure.Compute Emulator	Azure Compute Emulator	16.1.28810.153	Recommended
Microsoft.VisualStudio.Component.Azure.Storage.Emulator	Azure Storage Emulator	16.4.29313.120	Recommended
Microsoft.VisualStudio.Component.CloudExplorer	Cloud Explorer	16.0.28625.61	Recommended
Microsoft.VisualStudio.Component.DiagnosticTools	.NET profiling tools	16.4.29429.68	Recommended
Microsoft.VisualStudio.Component.Web	ASP.NET and web development tools	16.0.28517.75	Recommended
Microsoft.VisualStudio.Component.WebDeploy	Web Deploy	16.0.28517.75	Recommended
Microsoft.VisualStudio.ComponentGroup.AzureFunctions	Azure WebJobs Tools	16.0.28621.142	Recommended
Microsoft.VisualStudio.ComponentGroup.Web.CloudTools	Cloud tools for web development	16.2.29003.222	Recommended
Microsoft.VisualStudio.Component.Windows10SDK.18362	Windows 10 SDK (10.0.18362.0)	16.1.28829.92	Optional
Microsoft.VisualStudio.ComponentGroup.IISDevelopment	Development time IIS support	16.0.28315.86	Optional
Microsoft.VisualStudio.ComponentGroup.MSIX.Packaging	MSIX Packaging Tools	16.4.29409.204	Optional

Mobile development with .NET

ID: Microsoft.VisualStudio.Workload.NetCrossPlat

Description: Build cross-platform applications for iOS, Android or Windows using Xamarin.

Components included by this workload

COMPONENT ID	NAME	VERSION	DEPENDENCY TYPE
Component.OpenJDK	OpenJDK (Microsoft distribution)	16.1.28811.260	Required
Component.Xamarin	Xamarin	16.4.29409.204	Required
Component.Xamarin.RemotedSimulator	Xamarin Remoted Simulator	16.0.28315.86	Required

Component ID	Name	Version	Dependency Type
Microsoft.Component.MSBuild	MSBuild	16.4.29429.68	Required
Microsoft.Net.Component.4.6.1.TargetingPack	.NET Framework 4.6.1 targeting pack	16.0.28517.75	Required
Microsoft.Net.Component.4.7.2.TargetingPack	.NET Framework 4.7.2 targeting pack	16.0.28517.75	Required
Microsoft.Net.Component.4.8.SDK	.NET Framework 4.8 SDK	16.4.29313.120	Required
Microsoft.Net.ComponentGroup.DevelopmentPrerequisites	.NET Framework 4.7.2 development tools	16.3.29207.166	Required
Microsoft.NetCore.Component.DevelopmentTools	.NET Core development tools	16.4.29511.114	Required
Microsoft.NetCore.Component.SDK	.NET Core 3.1 SDK	16.4.29519.181	Required
Microsoft.VisualStudio.Component.FSharp	F# language support	16.0.28315.86	Required
Microsoft.VisualStudio.Component.IntelliCode	IntelliCode	0.1	Required
Microsoft.VisualStudio.Component.Merq	Common Xamarin internal tools	16.2.29012.281	Required
Microsoft.VisualStudio.Component.MonoDebugger	Mono debugger	16.0.28517.75	Required
Microsoft.VisualStudio.Component.NuGet	NuGet package manager	16.1.28829.92	Required
Microsoft.VisualStudio.Component.Roslyn.Compiler	C# and Visual Basic Roslyn compilers	16.0.28714.129	Required
Microsoft.VisualStudio.Component.Roslyn.LanguageServices	C# and Visual Basic	16.4.29429.68	Required
Microsoft.VisualStudio.ComponentGroup.WebToolsExtensions.TemplateEngine	ASP.NET templating engine	16.0.28315.86	Required
Component.Android.SDK28	Android SDK setup (API level 28)	16.2.29003.222	Recommended

ASP.NET and web development

ID: Microsoft.VisualStudio.Workload.NetWeb

Description: Build web applications using ASP.NET Core, ASP.NET, HTML/JavaScript, and Containers including Docker support.

Components included by this workload

Component ID	Name	Version	Dependency Type
Component.Microsoft.VisualStudio.RazorExtension	Razor Language Services	16.0.28714.129	Required
Component.Microsoft.Web.LibraryManager	Library Manager	16.0.28315.86	Required
Microsoft.Component.MSBuild	MSBuild	16.4.29429.68	Required
Microsoft.Net.Component.4.5.2.TargetingPack	.NET Framework 4.5.2 targeting pack	16.0.28517.75	Required
Microsoft.Net.Component.4.5.TargetingPack	.NET Framework 4.5 targeting pack	16.0.28517.75	Required
Microsoft.Net.Component.4.6.1.TargetingPack	.NET Framework 4.6.1 targeting pack	16.0.28517.75	Required
Microsoft.Net.Component.4.7.2.TargetingPack	.NET Framework 4.7.2 targeting pack	16.0.28517.75	Required
Microsoft.Net.Component.4.8.SDK	.NET Framework 4.8 SDK	16.4.29313.120	Required
Microsoft.Net.ComponentGroup.DevelopmentPrerequisites	.NET Framework 4.7.2 development tools	16.3.29207.166	Required
Microsoft.NetCore.Component.DevelopmentTools	.NET Core development tools	16.4.29511.114	Required
Microsoft.NetCore.Component.SDK	.NET Core 3.1 SDK	16.4.29519.181	Required
Microsoft.NetCore.Component.Web	.NET Core development tools	16.4.29511.114	Required
Microsoft.VisualStudio.Component.Common.Azure.Tools	Connectivity and publishing tools	16.4.29409.204	Required
Microsoft.VisualStudio.Component.DockerTools	Container development tools	16.4.29409.204	Required
Microsoft.VisualStudio.Component.FSharp	F# language support	16.0.28315.86	Required
Microsoft.VisualStudio.Component.FSharp.WebTemplates	F# language support for web projects	16.3.29207.166	Required

Component ID	Name	Version	Dependency Type
Microsoft.VisualStudio.Component.IISExpress	IIS Express	16.0.28315.86	Required
Microsoft.VisualStudio.Component.IntelliCode	IntelliCode	0.1	Required
Microsoft.VisualStudio.Component.JavaScript.Diagnostics	JavaScript diagnostics	16.0.28517.75	Required
Microsoft.VisualStudio.Component.JavaScript.TypeScript	JavaScript and TypeScript language support	16.4.29409.204	Required
Microsoft.VisualStudio.Component.ManagedDesktop.Core	Managed Desktop Workload Core	16.4.29318.151	Required
Microsoft.VisualStudio.Component.MSODBC.SQL	SQL Server ODBC Driver	16.0.28625.61	Required
Microsoft.VisualStudio.Component.MSSQL.CMDLnUtils	SQL Server Command Line Utilities	16.0.28707.177	Required
Microsoft.VisualStudio.Component.NuGet	NuGet package manager	16.1.28829.92	Required
Microsoft.VisualStudio.Component.Roslyn.Compiler	C# and Visual Basic Roslyn compilers	16.0.28714.129	Required
Microsoft.VisualStudio.Component.Roslyn.LanguageServices	C# and Visual Basic	16.4.29429.68	Required
Microsoft.VisualStudio.Component.SQL.ADAL	SQL ADAL runtime	16.0.28517.75	Required
Microsoft.VisualStudio.Component.SQL.CLR	CLR data types for SQL Server	16.0.28315.86	Required
Microsoft.VisualStudio.Component.SQL.DataSources	Data sources for SQL Server support	16.0.28315.86	Required
Microsoft.VisualStudio.Component.SQL.LocalDB.Runtime	SQL Server Express 2016 LocalDB	16.0.28625.61	Required
Microsoft.VisualStudio.Component.SQL.SSDT	SQL Server Data Tools	16.3.29207.166	Required
Microsoft.VisualStudio.Component.TextTemplating	Text Template Transformation	16.0.28625.61	Required
Microsoft.VisualStudio.Component.TypeScript.3.7	TypeScript 3.7 SDK	16.0.29429.68	Required

Component ID	Name	Version	Dependency Type
Microsoft.VisualStudio.Component.Web	ASP.NET and web development tools	16.0.28517.75	Required
Microsoft.VisualStudio.ComponentGroup.Web	ASP.NET and web development tools prerequisites	16.4.29318.151	Required
Microsoft.VisualStudio.ComponentGroup.WebToolsExtensions	ASP.NET and web development	16.0.28621.142	Required
Component.Microsoft.VisualStudio.LiveShare	Live Share	1.0.1076	Recommended
Component.Microsoft.VisualStudio.Web.AzureFunctions	Azure WebJobs Tools	16.0.28714.129	Recommended
Microsoft.Net.Component.4.5.1.TargetingPack	.NET Framework 4.5.1 targeting pack	16.0.28517.75	Recommended
Microsoft.Net.Component.4.6.TargetingPack	.NET Framework 4.6 targeting pack	16.0.28517.75	Recommended
Microsoft.Net.Component.4.TargetingPack	.NET Framework 4 targeting pack	16.0.28517.75	Recommended
Microsoft.Net.ComponentGroup.TargetingPacks.Common	.NET Framework 4 – 4.6 development tools	16.0.28516.191	Recommended
Microsoft.Net.Core.Component.SDK.2.1	.NET Core 2.1 LTS Runtime	16.4.29519.181	Recommended
Microsoft.VisualStudio.Component.AppInsights.Tools	Developer Analytics tools	16.4.29429.68	Recommended
Microsoft.VisualStudio.Component.AspNet45	Advanced ASP.NET features	16.0.28315.86	Recommended
Microsoft.VisualStudio.Component.Azure.AuthoringTools	Azure Authoring Tools	16.0.28625.61	Recommended
Microsoft.VisualStudio.Component.Azure.ClientLibs	Azure libraries for .NET	16.0.28315.86	Recommended
Microsoft.VisualStudio.Component.Azure.Compute.Emulator	Azure Compute Emulator	16.1.28810.153	Recommended
Microsoft.VisualStudio.Component.Azure.Storage.Emulator	Azure Storage Emulator	16.4.29313.120	Recommended

Component ID	Name	Version	Dependency Type
Microsoft.VisualStudio.Component.CloudExplorer	Cloud Explorer	16.0.28625.61	Recommended
Microsoft.VisualStudio.Component.DiagnosticTools	.NET profiling tools	16.4.29429.68	Recommended
Microsoft.VisualStudio.Component.EntityFramework	Entity Framework 6 tools	16.0.28315.86	Recommended
Microsoft.VisualStudio.Component.WebDeploy	Web Deploy	16.0.28517.75	Recommended
Microsoft.VisualStudio.ComponentGroup.AzureFunctions	Azure WebJobs Tools	16.0.28621.142	Recommended
Microsoft.VisualStudio.ComponentGroup.Web.CloudTools	Cloud tools for web development	16.2.29003.222	Recommended
Microsoft.Net.Component.4.6.2.TargetingPack	.NET Framework 4.6.2 targeting pack	16.0.28517.75	Optional
Microsoft.Net.Component.4.7.1.TargetingPack	.NET Framework 4.7.1 targeting pack	16.0.28517.75	Optional
Microsoft.Net.Component.4.7.TargetingPack	.NET Framework 4.7 targeting pack	16.0.28517.75	Optional
Microsoft.Net.Component.4.8.TargetingPack	.NET Framework 4.8 targeting pack	16.4.29313.120	Optional
Microsoft.Net.ComponentGroup.4.6.1.DeveloperTools	.NET Framework 4.6.1 development tools	16.3.29207.166	Optional
Microsoft.Net.ComponentGroup.4.6.2.DeveloperTools	.NET Framework 4.6.2 development tools	16.3.29207.166	Optional
Microsoft.Net.ComponentGroup.4.7.1.DeveloperTools	.NET Framework 4.7.1 development tools	16.3.29207.166	Optional
Microsoft.Net.ComponentGroup.4.7.DeveloperTools	.NET Framework 4.7 development tools	16.3.29207.166	Optional
Microsoft.Net.ComponentGroup.4.8.DeveloperTools	.NET Framework 4.8 development tools	16.4.29318.151	Optional
Microsoft.VisualStudio.Component.Wcf.Tooling	Windows Communication Foundation	16.0.28625.61	Optional
Microsoft.VisualStudio.ComponentGroup.AdditionalWebProjectTemplates	Additional project templates (previous versions)	16.0.28621.142	Optional
Microsoft.VisualStudio.ComponentGroup.IISDevelopment	Development time IIS support	16.0.28315.86	Optional

Node.js development

ID: Microsoft.VisualStudio.Workload.Node

Description: Build scalable network applications using Node.js, an asynchronous event-driven JavaScript runtime.

Components included by this workload

Component ID	Name	Version	Dependency Type
Microsoft.VisualStudio.Component.JavaScript.Diagnostics	JavaScript diagnostics	16.0.28517.75	Required
Microsoft.VisualStudio.Component.JavaScript.TypeScript	JavaScript and TypeScript language support	16.4.29409.204	Required
Microsoft.VisualStudio.Component.Node.Tools	Node.js development tools	16.4.29429.68	Required
Microsoft.VisualStudio.Component.TypeScript.3.7	TypeScript 3.7 SDK	16.0.29429.68	Required
Microsoft.VisualStudio.ComponentGroup.WebToolsExtensions	ASP.NET and web development	16.0.28621.142	Required
Component.Microsoft.VisualStudio.LiveShare	Live Share	1.0.1076	Recommended
Microsoft.VisualStudio.Component.IntelliCode	IntelliCode	0.1	Recommended
Microsoft.VisualStudio.Component.WebDeploy	Web Deploy	16.0.28517.75	Recommended
Microsoft.VisualStudio.Component.AppInsights.Tools	Developer Analytics tools	16.4.29429.68	Optional
Microsoft.VisualStudio.Component.Common.Azure.Tools	Connectivity and publishing tools	16.4.29409.204	Optional
Microsoft.VisualStudio.Component.VC.Corelde	C++ core features	16.0.28625.61	Optional
Microsoft.VisualStudio.Component.VC.Tools.x86.x64	MSVC v142 - VS 2019 C++ x64/x86 build tools (v14.24)	16.4.29409.204	Optional

Office/SharePoint development

ID: Microsoft.VisualStudio.Workload.Office

Description: Create Office and SharePoint add-ins, SharePoint solutions, and VSTO add-ins using C#, VB, and JavaScript.

Components included by this workload

Component ID	Name	Version	Dependency Type
Component.Microsoft.VisualStudio.RazorExtension	Razor Language Services	16.0.28714.129	Required
Component.Microsoft.Web.LibraryManager	Library Manager	16.0.28315.86	Required
Microsoft.Component.MSBuild	MSBuild	16.4.29429.68	Required
Microsoft.Net.Component.4.5.2.TargetingPack	.NET Framework 4.5.2 targeting pack	16.0.28517.75	Required
Microsoft.Net.Component.4.5.TargetingPack	.NET Framework 4.5 targeting pack	16.0.28517.75	Required
Microsoft.Net.Component.4.6.1.TargetingPack	.NET Framework 4.6.1 targeting pack	16.0.28517.75	Required
Microsoft.Net.Component.4.7.2.TargetingPack	.NET Framework 4.7.2 targeting pack	16.0.28517.75	Required
Microsoft.Net.Component.4.8.SDK	.NET Framework 4.8 SDK	16.4.29313.120	Required
Microsoft.Net.Component.4.TargetingPack	.NET Framework 4 targeting pack	16.0.28517.75	Required
Microsoft.Net.ComponentGroup.DevelopmentPrerequisites	.NET Framework 4.7.2 development tools	16.3.29207.166	Required
Microsoft.NetCore.Component.SDK	.NET Core 3.1 SDK	16.4.29519.181	Required
Microsoft.VisualStudio.Component.AppInsights.Tools	Developer Analytics tools	16.4.29429.68	Required
Microsoft.VisualStudio.Component.Common.Azure.Tools	Connectivity and publishing tools	16.4.29409.204	Required
Microsoft.VisualStudio.Component.DockerTools	Container development tools	16.4.29409.204	Required
Microsoft.VisualStudio.Component.IISExpress	IIS Express	16.0.28315.86	Required
Microsoft.VisualStudio.Component.IntelliCode	IntelliCode	0.1	Required
Microsoft.VisualStudio.Component.JavaScript.Diagnostics	JavaScript diagnostics	16.0.28517.75	Required
Microsoft.VisualStudio.Component.JavaScript.TypeScript	JavaScript and TypeScript language support	16.4.29409.204	Required

Component ID	Name	Version	Dependency Type
Microsoft.VisualStudio.Component.ManagedDesktop.Core	Managed Desktop Workload Core	16.4.29318.151	Required
Microsoft.VisualStudio.Component.ManagedDesktop.Prerequisites	.NET desktop development tools	16.4.29429.68	Required
Microsoft.VisualStudio.Component.MSODBC.SQL	SQL Server ODBC Driver	16.0.28625.61	Required
Microsoft.VisualStudio.Component.MSSQL.CMDLnUtils	SQL Server Command Line Utilities	16.0.28707.177	Required
Microsoft.VisualStudio.Component.NuGet	NuGet package manager	16.1.28829.92	Required
Microsoft.VisualStudio.Component.Roslyn.Compiler	C# and Visual Basic Roslyn compilers	16.0.28714.129	Required
Microsoft.VisualStudio.Component.Roslyn.LanguageServices	C# and Visual Basic	16.4.29429.68	Required
Microsoft.VisualStudio.Component.Sharepoint.Tools	Office Developer Tools for Visual Studio	16.4.29409.204	Required
Microsoft.VisualStudio.Component.SQL.ADAL	SQL ADAL runtime	16.0.28517.75	Required
Microsoft.VisualStudio.Component.SQL.CLR	CLR data types for SQL Server	16.0.28315.86	Required
Microsoft.VisualStudio.Component.SQL.DataSources	Data sources for SQL Server support	16.0.28315.86	Required
Microsoft.VisualStudio.Component.SQL.LocalDB.Runtime	SQL Server Express 2016 LocalDB	16.0.28625.61	Required
Microsoft.VisualStudio.Component.SQL.SSDT	SQL Server Data Tools	16.3.29207.166	Required
Microsoft.VisualStudio.Component.TextTemplating	Text Template Transformation	16.0.28625.61	Required
Microsoft.VisualStudio.Component.TypeScript.3.7	TypeScript 3.7 SDK	16.0.29429.68	Required
Microsoft.VisualStudio.Component.Wcf.Tooling	Windows Communication Foundation	16.0.28625.61	Required
Microsoft.VisualStudio.Component.Web	ASP.NET and web development tools	16.0.28517.75	Required

Component ID	Name	Version	Dependency Type
Microsoft.VisualStudio.Component.Workflow	Windows Workflow Foundation	16.0.28315.86	Required
Microsoft.VisualStudio.ComponentGroup.Web	ASP.NET and web development tools prerequisites	16.4.29318.151	Required
Microsoft.VisualStudio.ComponentGroup.WebToolsExtensions	ASP.NET and web development	16.0.28621.142	Required
Microsoft.VisualStudio.Component.TeamOffice	Visual Studio Tools for Office (VSTO)	16.4.29409.204	Recommended
Microsoft.VisualStudio.Component.WebDeploy	Web Deploy	16.0.28517.75	Recommended
Microsoft.Net.Component.4.6.2.TargetingPack	.NET Framework 4.6.2 targeting pack	16.0.28517.75	Optional
Microsoft.Net.Component.4.7.1.TargetingPack	.NET Framework 4.7.1 targeting pack	16.0.28517.75	Optional
Microsoft.Net.Component.4.7.TargetingPack	.NET Framework 4.7 targeting pack	16.0.28517.75	Optional
Microsoft.Net.Component.4.8.TargetingPack	.NET Framework 4.8 targeting pack	16.4.29313.120	Optional
Microsoft.Net.ComponentGroup.4.6.1.DeveloperTools	.NET Framework 4.6.1 development tools	16.3.29207.166	Optional
Microsoft.Net.ComponentGroup.4.6.2.DeveloperTools	.NET Framework 4.6.2 development tools	16.3.29207.166	Optional
Microsoft.Net.ComponentGroup.4.7.1.DeveloperTools	.NET Framework 4.7.1 development tools	16.3.29207.166	Optional
Microsoft.Net.ComponentGroup.4.7.DeveloperTools	.NET Framework 4.7 development tools	16.3.29207.166	Optional
Microsoft.Net.ComponentGroup.4.8.DeveloperTools	.NET Framework 4.8 development tools	16.4.29318.151	Optional
Microsoft.VisualStudio.ComponentGroup.Sharepoint.WIF	Windows Identity Foundation 3.5	16.0.28621.142	Optional

Python development

ID: Microsoft.VisualStudio.Workload.Python

Description: Editing, debugging, interactive development and source control for Python.

Components included by this workload

Component ID	Name	Version	Dependency Type
Microsoft.Component.PythonTools	Python language support	16.4.29429.68	Required
Component.CPython3.x64	Python 3 64-bit (3.7.5)	3.7.5	Recommended
Component.Microsoft.VisualStudio.LiveShare	Live Share	1.0.1076	Recommended
Microsoft.Component.PythonTools.Minicondax64	Python miniconda	16.2.29003.222	Recommended
Microsoft.Component.PythonTools.Web	Python web support	16.0.28517.75	Recommended
Microsoft.VisualStudio.Component.Common.Azure.Tools	Connectivity and publishing tools	16.4.29409.204	Recommended
Microsoft.VisualStudio.Component.JavaScript.TypeScript	JavaScript and TypeScript language support	16.4.29409.204	Recommended
Microsoft.VisualStudio.Component.TypeScript.3.7	TypeScript 3.7 SDK	16.0.29429.68	Recommended
Microsoft.VisualStudio.Component.WebDeploy	Web Deploy	16.0.28517.75	Recommended
Microsoft.VisualStudio.ComponentGroup.WebToolsExtensions	ASP.NET and web development	16.0.28621.142	Recommended
Component.CPython2.x64	Python 2 64-bit (2.7.16)	2.7.16	Optional
Component.CPython2.x86	Python 2 32-bit (2.7.16)	2.7.16	Optional
Component.CPython3.x86	Python 3 32-bit (3.7.5)	3.7.5	Optional
Component.Microsoft.VisualStudio.RazorExtension	Razor Language Services	16.0.28714.129	Optional
Component.Microsoft.Web.LibraryManager	Library Manager	16.0.28315.86	Optional
Microsoft.Component.MSBuild	MSBuild	16.4.29429.68	Optional
Microsoft.ComponentGroup.PythonTools.NativeDevelopment	Python native development tools	16.2.29020.229	Optional
Microsoft.Net.Component.4.5.2.TargetingPack	.NET Framework 4.5.2 targeting pack	16.0.28517.75	Optional

Component ID	Name	Version	Dependency Type
Microsoft.Net.Component.4.5.TargetingPack	.NET Framework 4.5 targeting pack	16.0.28517.75	Optional
Microsoft.Net.Component.4.7.2.TargetingPack	.NET Framework 4.7.2 targeting pack	16.0.28517.75	Optional
Microsoft.Net.Component.4.8.SDK	.NET Framework 4.8 SDK	16.4.29313.120	Optional
Microsoft.Net.ComponentGroup.DevelopmentPrerequisites	.NET Framework 4.7.2 development tools	16.3.29207.166	Optional
Microsoft.NetCore.Component.SDK	.NET Core 3.1 SDK	16.4.29519.181	Optional
Microsoft.VisualStudio.Component.Azure.AuthoringTools	Azure Authoring Tools	16.0.28625.61	Optional
Microsoft.VisualStudio.Component.Azure.ClientLibs	Azure libraries for .NET	16.0.28315.86	Optional
Microsoft.VisualStudio.Component.Azure.Compute.Emulator	Azure Compute Emulator	16.1.28810.153	Optional
Microsoft.VisualStudio.Component.Azure.Storage.Emulator	Azure Storage Emulator	16.4.29313.120	Optional
Microsoft.VisualStudio.Component.Azure.Waverton	Azure Cloud Services core tools	16.4.29409.204	Optional
Microsoft.VisualStudio.Component.Azure.Waverton.BuildTools	Azure Cloud Services build tools	16.3.29207.166	Optional
Microsoft.VisualStudio.Component.DockerTools	Container development tools	16.4.29409.204	Optional
Microsoft.VisualStudio.Component.Graphics.Tools	Graphics debugger and GPU profiler for DirectX	16.0.28625.61	Optional
Microsoft.VisualStudio.Component.IISExpress	IIS Express	16.0.28315.86	Optional
Microsoft.VisualStudio.Component.JavaScript.Diagnostics	JavaScript diagnostics	16.0.28517.75	Optional
Microsoft.VisualStudio.Component.ManagedDesktop.Core	Managed Desktop Workload Core	16.4.29318.151	Optional

Component ID	Name	Version	Dependency Type
Microsoft.VisualStudio.Component.MSODBC.SQL	SQL Server ODBC Driver	16.0.28625.61	Optional
Microsoft.VisualStudio.Component.MSSQL.CMDLnUtils	SQL Server Command Line Utilities	16.0.28707.177	Optional
Microsoft.VisualStudio.Component.NuGet	NuGet package manager	16.1.28829.92	Optional
Microsoft.VisualStudio.Component.Roslyn.Compiler	C# and Visual Basic Roslyn compilers	16.0.28714.129	Optional
Microsoft.VisualStudio.Component.Roslyn.LanguageServices	C# and Visual Basic	16.4.29429.68	Optional
Microsoft.VisualStudio.Component.SQL.ADAL	SQL ADAL runtime	16.0.28517.75	Optional
Microsoft.VisualStudio.Component.SQL.CLR	CLR data types for SQL Server	16.0.28315.86	Optional
Microsoft.VisualStudio.Component.SQL.DataSources	Data sources for SQL Server support	16.0.28315.86	Optional
Microsoft.VisualStudio.Component.SQL.LocalDB.Runtime	SQL Server Express 2016 LocalDB	16.0.28625.61	Optional
Microsoft.VisualStudio.Component.SQL.SSDT	SQL Server Data Tools	16.3.29207.166	Optional
Microsoft.VisualStudio.Component.TextTemplating	Text Template Transformation	16.0.28625.61	Optional
Microsoft.VisualStudio.Component.VC.Coreide	C++ core features	16.0.28625.61	Optional
Microsoft.VisualStudio.Component.VC.DiagnosticTools	C++ profiling tools	16.4.29429.68	Optional
Microsoft.VisualStudio.Component.VC.Tools.x86.x64	MSVC v142 - VS 2019 C++ x64/x86 build tools (v14.24)	16.4.29409.204	Optional
Microsoft.VisualStudio.Component.Web	ASP.NET and web development tools	16.0.28517.75	Optional
Microsoft.VisualStudio.Component.Windows10SDK	Windows Universal C Runtime	16.4.29409.204	Optional
Microsoft.VisualStudio.Component.Windows10SDK.18362	Windows 10 SDK (10.0.18362.0)	16.1.28829.92	Optional

Component ID	Name	Version	Dependency Type
Microsoft.VisualStudio.ComponentGroup.Web	ASP.NET and web development tools prerequisites	16.4.29318.151	Optional

Universal Windows Platform development

ID: Microsoft.VisualStudio.Workload.Universal

Description: Create applications for the Universal Windows Platform with C#, VB, or optionally C++.

Components included by this workload

Component ID	Name	Version	Dependency Type
Microsoft.Component.NetFX.Native	.NET Native	16.4.29429.68	Required
Microsoft.ComponentGroup.Blend	Blend for Visual Studio	16.0.28315.86	Required
Microsoft.Net.Component.4.5.TargetingPack	.NET Framework 4.5 targeting pack	16.0.28517.75	Required
Microsoft.NetCore.Component.SDK	.NET Core 3.1 SDK	16.4.29519.181	Required
Microsoft.VisualStudio.Component.AppInsights.Tools	Developer Analytics tools	16.4.29429.68	Required
Microsoft.VisualStudio.Component.DiagnosticTools	.NET profiling tools	16.4.29429.68	Required
Microsoft.VisualStudio.Component.Graphics	Image and 3D model editors	16.0.28517.75	Required
Microsoft.VisualStudio.Component.IntelliCode	IntelliCode	0.1	Required
Microsoft.VisualStudio.Component.NuGet	NuGet package manager	16.1.28829.92	Required
Microsoft.VisualStudio.Component.Roslyn.Compiler	C# and Visual Basic Roslyn compilers	16.0.28714.129	Required
Microsoft.VisualStudio.Component.Roslyn.LanguageServices	C# and Visual Basic	16.4.29429.68	Required
Microsoft.VisualStudio.Component.SQL.CLR	CLR data types for SQL Server	16.0.28315.86	Required
Microsoft.VisualStudio.Component.Windows10SDK.18362	Windows 10 SDK (10.0.18362.0)	16.1.28829.92	Required

Component ID	Name	Version	Dependency Type
Microsoft.VisualStudio.ComponentGroup.MSIX.Packaging	MSIX Packaging Tools	16.4.29409.204	Required
Microsoft.VisualStudio.ComponentGroup.UWP.NetCoreAndStandard	.NET Native and .NET Standard	16.3.29102.218	Required
Microsoft.VisualStudio.ComponentGroup.UWP.Support	Universal Windows Platform tools	16.4.29409.204	Required
Microsoft.VisualStudio.ComponentGroup.UWP.Xamarin	Universal Windows Platform tools for Xamarin	16.4.29511.114	Required
Microsoft.Component.MSBuild	MSBuild	16.4.29429.68	Optional
Microsoft.Net.Component.4.8.SDK	.NET Framework 4.8 SDK	16.4.29313.120	Optional
Microsoft.VisualStudio.Component.Graphics.Tools	Graphics debugger and GPU profiler for DirectX	16.0.28625.61	Optional
Microsoft.VisualStudio.Component.TextTemplating	Text Template Transformation	16.0.28625.61	Optional
Microsoft.VisualStudio.Component.UWP.VC.ARM64	C++ Universal Windows Platform support for v142 build tools (ARM64)	16.3.29207.166	Optional
Microsoft.VisualStudio.Component.VC.CoreLde	C++ core features	16.0.28625.61	Optional
Microsoft.VisualStudio.Component.VC.Redist.14.Latest	C++ 2019 Redistributable Update	16.4.29429.68	Optional
Microsoft.VisualStudio.Component.VC.Tools.ARM	MSVC v142 - VS 2019 C++ ARM build tools (v14.24)	16.4.29409.204	Optional
Microsoft.VisualStudio.Component.VC.Tools.ARM64	MSVC v142 - VS 2019 C++ ARM64 build tools (v14.24)	16.4.29409.204	Optional
Microsoft.VisualStudio.Component.VC.Tools.x86.x64	MSVC v142 - VS 2019 C++ x64/x86 build tools (v14.24)	16.4.29409.204	Optional
Microsoft.VisualStudio.Component.VC.v141.ARM	MSVC v141 - VS 2017 C++ ARM build tools (v14.16)	16.2.29003.222	Optional
Microsoft.VisualStudio.Component.VC.v141.ARM64	MSVC v141 - VS 2017 C++ ARM64 build tools (v14.16)	16.1.28829.92	Optional
Microsoft.VisualStudio.Component.VC.v141.x86.x64	MSVC v141 - VS 2017 C++ x64/x86 build tools (v14.16)	16.1.28829.92	Optional

Component ID	Name	Version	Dependency Type
Microsoft.VisualStudio.Component.Windows10SDK.16299	Windows 10 SDK (10.0.16299.0)	16.0.28517.75	Optional
Microsoft.VisualStudio.Component.Windows10SDK.17134	Windows 10 SDK (10.0.17134.0)	16.0.28517.75	Optional
Microsoft.VisualStudio.Component.Windows10SDK.17763	Windows 10 SDK (10.0.17763.0)	16.0.28517.75	Optional
Microsoft.VisualStudio.Component.Windows10SDK.IoPOverUsb	USB Device Connectivity	16.4.29511.114	Optional
Microsoft.VisualStudio.ComponentGroup.NativeDesktop.Core	C++ core desktop features	16.2.29012.281	Optional
Microsoft.VisualStudio.ComponentGroup.UWP.VC	C++ (v142) Universal Windows Platform tools	16.3.29207.166	Optional
Microsoft.VisualStudio.ComponentGroup.UWP.VC.v141	C++ (v141) Universal Windows Platform tools	16.1.28810.153	Optional

Visual Studio extension development

ID: Microsoft.VisualStudio.Workload.VisualStudioExtension

Description: Create add-ons and extensions for Visual Studio, including new commands, code analyzers and tool windows.

Components included by this workload

Component ID	Name	Version	Dependency Type
Microsoft.Component.MSBuild	MSBuild	16.4.29429.68	Required
Microsoft.Net.Component.4.6.TargetingPack	.NET Framework 4.6 targeting pack	16.0.28517.75	Required
Microsoft.Net.Component.4.7.2.TargetingPack	.NET Framework 4.7.2 targeting pack	16.0.28517.75	Required
Microsoft.Net.Component.4.8.SDK	.NET Framework 4.8 SDK	16.4.29313.120	Required
Microsoft.Net.ComponentGroup.DevelopmentPrerequisites	.NET Framework 4.7.2 development tools	16.3.29207.166	Required

Component ID	Name	Version	Dependency Type
Microsoft.VisualStudio.Component.IntelliCode	IntelliCode	0.1	Required
Microsoft.VisualStudio.Component.NuGet	NuGet package manager	16.1.28829.92	Required
Microsoft.VisualStudio.Component.Roslyn.Compiler	C# and Visual Basic Roslyn compilers	16.0.28714.129	Required
Microsoft.VisualStudio.Component.Roslyn.LanguageServices	C# and Visual Basic	16.4.29429.68	Required
Microsoft.VisualStudio.Component.VSSDK	Visual Studio SDK	16.0.28315.86	Required
Microsoft.VisualStudio.ComponentGroup.VisualStudioExtension.Prerequisites	Visual Studio extension development prerequisites	16.4.29318.151	Required
Microsoft.VisualStudio.Component.DiagnosticTools	.NET profiling tools	16.4.29429.68	Recommended
Microsoft.VisualStudio.Component.TextTemplating	Text Template Transformation	16.0.28625.61	Recommended
Microsoft.Component.CodeAnalysis.SDK	.NET Compiler Platform SDK	16.2.29003.222	Optional
Microsoft.VisualStudio.Component.AppInsights.Tools	Developer Analytics tools	16.4.29429.68	Optional
Microsoft.VisualStudio.Component.DslTools	Modeling SDK	16.0.28315.86	Optional

Unaffiliated components

These are components that are not included with any workload, but may be selected as an individual component.

Component ID	Name	Version
Component.GitHub.VisualStudio	GitHub extension for Visual Studio	2.5.9.5485
Component.Xamarin.Inspector	Xamarin Inspector	16.0.28315.86
Component.Xamarin.Profiler	Xamarin Profiler	16.0.28315.86
Component.Xamarin.Workbooks	Xamarin Workbooks	16.0.28315.86
Microsoft.Component.ClickOnce	ClickOnce Publishing	16.4.29409.204
Microsoft.Component.HelpViewer	Help Viewer	16.0.28625.61

Component ID	Name	Version
Microsoft.Net.Component.4.6.1.SDK	.NET Framework 4.6.1 SDK	16.4.29409.204
Microsoft.Net.Component.4.6.2.SDK	.NET Framework 4.6.2 SDK	16.4.29409.204
Microsoft.Net.Component.4.7.1.SDK	.NET Framework 4.7.1 SDK	16.4.29409.204
Microsoft.Net.Component.4.7.2.SDK	.NET Framework 4.7.2 SDK	16.4.29409.204
Microsoft.Net.Component.4.7.SDK	.NET Framework 4.7 SDK	16.4.29409.204
Microsoft.Net.Core.Component.SDK.2.2	.NET Core 2.2 Runtime	16.4.29519.181
Microsoft.Net.Core.Component.SDK.3.0	.NET Core 3.0 Runtime	16.4.29519.181
Microsoft.NetCore.ComponentGroup.DevelopmentTools.2.1	Development Tools plus .NET Core 2.1	16.3.29207.166
Microsoft.NetCore.ComponentGroup.Web.2.1	Web Development Tools plus .NET Core 2.1	16.3.29207.166
Microsoft.VisualStudio.Component.AzureDevOps.OfficeIntegration	Azure DevOps Office Integration	16.0.28625.61
Microsoft.VisualStudio.Component.ClassDesigner	Class Designer	16.0.28528.71
Microsoft.VisualStudio.Component.DependencyValidation.Community	Dependency Validation	16.0.28517.75
Microsoft.VisualStudio.Component.Git	Git for Windows	16.0.28625.61
Microsoft.VisualStudio.Component.GraphDocument	DGML editor	16.0.28625.61
Microsoft.VisualStudio.Component.LinqToSql	LINQ to SQL tools	16.0.28625.61
Microsoft.VisualStudio.Component.VC.14.20.ARM	MSVC v142 - VS 2019 C++ ARM build tools (v14.20)	16.4.29511.114
Microsoft.VisualStudio.Component.VC.14.20.ARM.Spectre	MSVC v142 - VS 2019 C++ ARM Spectre-mitigated libs (v14.20)	16.4.29511.114
Microsoft.VisualStudio.Component.VC.14.20.ARM64	MSVC v142 - VS 2019 C++ ARM64 build tools (v14.20)	16.4.29511.114
Microsoft.VisualStudio.Component.VC.14.20.ARM64.Spectre	MSVC v142 - VS 2019 C++ ARM64 Spectre-mitigated libs (v14.20)	16.4.29511.114
Microsoft.VisualStudio.Component.VC.14.20.ATL	C++ v14.20 ATL for v142 build tools (x86 & x64)	16.1.28829.92

Component ID	Name	Version
Microsoft.VisualStudio.Component.VC.14.20.ATL.ARM	C++ v14.20 ATL for v142 build tools (ARM)	16.4.29511.114
Microsoft.VisualStudio.Component.VC.14.20.ATL.ARM.Spectre	C++ v14.20 ATL for v142 build tools with Spectre Mitigations (ARM)	16.4.29511.114
Microsoft.VisualStudio.Component.VC.14.20.ATL.ARM64	C++ v14.20 ATL for v142 build tools (ARM64)	16.4.29511.114
Microsoft.VisualStudio.Component.VC.14.20.ATL.ARM64.Spectre	C++ v14.20 ATL for v142 build tools with Spectre Mitigations (ARM64)	16.4.29511.114
Microsoft.VisualStudio.Component.VC.14.20.ATL.Spectre	C++ v14.20 ATL for v142 build tools with Spectre Mitigations (x86 & x64)	16.4.29511.114
Microsoft.VisualStudio.Component.VC.14.20.CLI.Support	C++/CLI support for v142 build tools (14.20)	16.4.29409.204
Microsoft.VisualStudio.Component.VC.14.20.MFC	C++ v14.20 MFC for v142 build tools (x86 & x64)	16.2.29003.222
Microsoft.VisualStudio.Component.VC.14.20.MFC.ARM	C++ v14.20 MFC for v142 build tools (ARM)	16.4.29511.114
Microsoft.VisualStudio.Component.VC.14.20.MFC.ARM.Spectre	C++ v14.20 MFC for v142 build tools with Spectre Mitigations (ARM)	16.4.29511.114
Microsoft.VisualStudio.Component.VC.14.20.MFC.ARM64	C++ v14.20 MFC for v142 build tools (ARM64)	16.4.29511.114
Microsoft.VisualStudio.Component.VC.14.20.MFC.ARM64.Spectre	C++ v14.20 MFC for v142 build tools with Spectre Mitigations (ARM64)	16.4.29511.114
Microsoft.VisualStudio.Component.VC.14.20.MFC.Spectre	C++ v14.20 MFC for v142 build tools with Spectre Mitigations (x86 & x64)	16.4.29511.114
Microsoft.VisualStudio.Component.VC.14.20.x86.x64	MSVC v142 - VS 2019 C++ x64/x86 build tools (v14.20)	16.4.29511.114
Microsoft.VisualStudio.Component.VC.14.20.x86.x64.Spectre	MSVC v142 - VS 2019 C++ x64/x86 Spectre-mitigated libs (v14.20)	16.4.29511.114
Microsoft.VisualStudio.Component.VC.14.21.ARM	MSVC v142 - VS 2019 C++ ARM build tools (v14.21)	16.3.29207.166
Microsoft.VisualStudio.Component.VC.14.21.ARM.Spectre	MSVC v142 - VS 2019 C++ ARM Spectre-mitigated libs (v14.21)	16.4.29429.68
Microsoft.VisualStudio.Component.VC.14.21.ARM64	MSVC v142 - VS 2019 C++ ARM64 build tools (v14.21)	16.3.29207.166
Microsoft.VisualStudio.Component.VC.14.21.ARM64.Spectre	MSVC v142 - VS 2019 C++ ARM64 Spectre-mitigated libs (v14.21)	16.4.29429.68

Component ID	Name	Version
Microsoft.VisualStudio.Component.VC.14.21.ATL	C++ v14.21 ATL for v142 build tools (x86 & x64)	16.2.29019.55
Microsoft.VisualStudio.Component.VC.14.21.ATL.ARM	C++ v14.21 ATL for v142 build tools (ARM)	16.2.29019.55
Microsoft.VisualStudio.Component.VC.14.21.ATL.ARM.Spectre	C++ v14.21 ATL for v142 build tools with Spectre Mitigations (ARM)	16.4.29429.68
Microsoft.VisualStudio.Component.VC.14.21.ATL.ARM64	C++ v14.21 ATL for v142 build tools (ARM64)	16.2.29019.55
Microsoft.VisualStudio.Component.VC.14.21.ATL.ARM64.Spectre	C++ v14.21 ATL for v142 build tools with Spectre Mitigations (ARM64)	16.4.29429.68
Microsoft.VisualStudio.Component.VC.14.21.ATL.Spectre	C++ v14.21 ATL for v142 build tools with Spectre Mitigations (x86 & x64)	16.4.29429.68
Microsoft.VisualStudio.Component.VC.14.21.CLI.Support	C++/CLI support for v142 build tools (14.21)	16.3.29207.166
Microsoft.VisualStudio.Component.VC.14.21.MFC	C++ v14.21 MFC for v142 build tools (x86 & x64)	16.2.29019.55
Microsoft.VisualStudio.Component.VC.14.21.MFC.ARM	C++ v14.21 MFC for v142 build tools (ARM)	16.2.29019.55
Microsoft.VisualStudio.Component.VC.14.21.MFC.ARM.Spectre	C++ v14.21 MFC for v142 build tools with Spectre Mitigations (ARM)	16.4.29429.68
Microsoft.VisualStudio.Component.VC.14.21.MFC.ARM64	C++ v14.21 MFC for v142 build tools (ARM64)	16.2.29019.55
Microsoft.VisualStudio.Component.VC.14.21.MFC.ARM64.Spectre	C++ v14.21 MFC for v142 build tools with Spectre Mitigations (ARM64)	16.4.29429.68
Microsoft.VisualStudio.Component.VC.14.21.MFC.Spectre	C++ v14.21 MFC for v142 build tools with Spectre Mitigations (x86 & x64)	16.4.29429.68
Microsoft.VisualStudio.Component.VC.14.21.x86.x64	MSVC v142 - VS 2019 C++ x64/x86 build tools (v14.21)	16.3.29207.166
Microsoft.VisualStudio.Component.VC.14.21.x86.x64.Spectre	MSVC v142 - VS 2019 C++ x64/x86 Spectre-mitigated libs (v14.21)	16.4.29429.68
Microsoft.VisualStudio.Component.VC.14.22.ARM	MSVC v142 - VS 2019 C++ ARM build tools (v14.22)	16.4.29313.120
Microsoft.VisualStudio.Component.VC.14.22.ARM.Spectre	MSVC v142 - VS 2019 C++ ARM Spectre-mitigated libs (v14.22)	16.4.29429.68
Microsoft.VisualStudio.Component.VC.14.22.ARM64	MSVC v142 - VS 2019 C++ ARM64 build tools (v14.22)	16.4.29313.120

Component ID	Name	Version
Microsoft.VisualStudio.Component.VC.14.22.ARM64.Spectre	MSVC v142 - VS 2019 C++ ARM64 Spectre-mitigated libs (v14.22)	16.4.29429.68
Microsoft.VisualStudio.Component.VC.14.22.ATL	C++ v14.22 ATL for v142 build tools (x86 & x64)	16.4.29313.120
Microsoft.VisualStudio.Component.VC.14.22.ATL.ARM	C++ v14.22 ATL for v142 build tools (ARM)	16.4.29313.120
Microsoft.VisualStudio.Component.VC.14.22.ATL.ARM.Spectre	C++ v14.22 ATL for v142 build tools with Spectre Mitigations (ARM)	16.4.29429.68
Microsoft.VisualStudio.Component.VC.14.22.ATL.ARM64	C++ v14.22 ATL for v142 build tools (ARM64)	16.4.29313.120
Microsoft.VisualStudio.Component.VC.14.22.ATL.ARM64.Spectre	C++ v14.22 ATL for v142 build tools with Spectre Mitigations (ARM64)	16.4.29429.68
Microsoft.VisualStudio.Component.VC.14.22.ATL.Spectre	C++ v14.22 ATL for v142 build tools with Spectre Mitigations (x86 & x64)	16.4.29429.68
Microsoft.VisualStudio.Component.VC.14.22.CLI.Support	C++/CLI support for v142 build tools (14.22)	16.4.29313.120
Microsoft.VisualStudio.Component.VC.14.22.MFC	C++ v14.22 MFC for v142 build tools (x86 & x64)	16.4.29313.120
Microsoft.VisualStudio.Component.VC.14.22.MFC.ARM	C++ v14.22 MFC for v142 build tools (ARM)	16.4.29313.120
Microsoft.VisualStudio.Component.VC.14.22.MFC.ARM.Spectre	C++ v14.22 MFC for v142 build tools with Spectre Mitigations (ARM)	16.4.29429.68
Microsoft.VisualStudio.Component.VC.14.22.MFC.ARM64	C++ v14.22 MFC for v142 build tools (ARM64)	16.4.29313.120
Microsoft.VisualStudio.Component.VC.14.22.MFC.ARM64.Spectre	C++ v14.22 MFC for v142 build tools with Spectre Mitigations (ARM64)	16.4.29429.68
Microsoft.VisualStudio.Component.VC.14.22.MFC.Spectre	C++ v14.22 MFC for v142 build tools with Spectre Mitigations (x86 & x64)	16.4.29429.68
Microsoft.VisualStudio.Component.VC.14.22.x86.x64	MSVC v142 - VS 2019 C++ x64/x86 build tools (v14.22)	16.4.29313.120
Microsoft.VisualStudio.Component.VC.14.22.x86.x64.Spectre	MSVC v142 - VS 2019 C++ x64/x86 Spectre-mitigated libs (v14.22)	16.4.29429.68
Microsoft.VisualStudio.Component.VC.14.23.ARM	MSVC v142 - VS 2019 C++ ARM build tools (v14.23)	16.4.29429.68
Microsoft.VisualStudio.Component.VC.14.23.ARM.Spectre	MSVC v142 - VS 2019 C++ ARM Spectre-mitigated libs (v14.23)	16.4.29429.68

Component ID	Name	Version
Microsoft.VisualStudio.Component.VC.14.23.ARM64	MSVC v142 - VS 2019 C++ ARM64 build tools (v14.23)	16.4.29429.68
Microsoft.VisualStudio.Component.VC.14.23.ARM64.Spectre	MSVC v142 - VS 2019 C++ ARM64 Spectre-mitigated libs (v14.23)	16.4.29429.68
Microsoft.VisualStudio.Component.VC.14.23.ATL	C++ v14.23 ATL for v142 build tools (x86 & x64)	16.4.29429.68
Microsoft.VisualStudio.Component.VC.14.23.ATL.ARM	C++ v14.23 ATL for v142 build tools (ARM)	16.4.29429.68
Microsoft.VisualStudio.Component.VC.14.23.ATL.ARM.Spectre	C++ v14.23 ATL for v142 build tools with Spectre Mitigations (ARM)	16.4.29429.68
Microsoft.VisualStudio.Component.VC.14.23.ATL.ARM64	C++ v14.23 ATL for v142 build tools (ARM64)	16.4.29429.68
Microsoft.VisualStudio.Component.VC.14.23.ATL.ARM64.Spectre	C++ v14.23 ATL for v142 build tools with Spectre Mitigations (ARM64)	16.4.29429.68
Microsoft.VisualStudio.Component.VC.14.23.ATL.Spectre	C++ v14.23 ATL for v142 build tools with Spectre Mitigations (x86 & x64)	16.4.29429.68
Microsoft.VisualStudio.Component.VC.14.23.CLI.Support	C++/CLI support for v142 build tools (14.23)	16.4.29409.204
Microsoft.VisualStudio.Component.VC.14.23.MFC	C++ v14.23 MFC for v142 build tools (x86 & x64)	16.4.29429.68
Microsoft.VisualStudio.Component.VC.14.23.MFC.ARM	C++ v14.23 MFC for v142 build tools (ARM)	16.4.29429.68
Microsoft.VisualStudio.Component.VC.14.23.MFC.ARM.Spectre	C++ v14.23 MFC for v142 build tools with Spectre Mitigations (ARM)	16.4.29429.68
Microsoft.VisualStudio.Component.VC.14.23.MFC.ARM64	C++ v14.23 MFC for v142 build tools (ARM64)	16.4.29429.68
Microsoft.VisualStudio.Component.VC.14.23.MFC.ARM64.Spectre	C++ v14.23 MFC for v142 build tools with Spectre Mitigations (ARM64)	16.4.29429.68
Microsoft.VisualStudio.Component.VC.14.23.MFC.Spectre	C++ v14.23 MFC for v142 build tools with Spectre Mitigations (x86 & x64)	16.4.29429.68
Microsoft.VisualStudio.Component.VC.14.23.x86.x64	MSVC v142 - VS 2019 C++ x64/x86 build tools (v14.23)	16.4.29429.68
Microsoft.VisualStudio.Component.VC.14.23.x86.x64.Spectre	MSVC v142 - VS 2019 C++ x64/x86 Spectre-mitigated libs (v14.23)	16.4.29429.68
Microsoft.VisualStudio.Component.VC.ATL.ARM	C++ ATL for latest v142 build tools (ARM)	16.4.29313.120

Component ID	Name	Version
Microsoft.VisualStudio.Component.VC.ATL.ARM.Spectre	C++ ATL for latest v142 build tools with Spectre Mitigations (ARM)	16.4.29429.68
Microsoft.VisualStudio.Component.VC.ATL.ARM64	C++ ATL for latest v142 build tools (ARM64)	16.4.29313.120
Microsoft.VisualStudio.Component.VC.ATL.ARM64.Spectre	C++ ATL for latest v142 build tools with Spectre Mitigations (ARM64)	16.4.29429.68
Microsoft.VisualStudio.Component.VC.ATL.Spectre	C++ ATL for latest v142 build tools with Spectre Mitigations (x86 & x64)	16.4.29429.68
Microsoft.VisualStudio.Component.VC.ATLMFC.Spectre	C++ MFC for latest v142 build tools with Spectre Mitigations (x86 & x64)	16.4.29429.68
Microsoft.VisualStudio.Component.VC.MFC.ARM	C++ MFC for latest v142 build tools (ARM)	16.4.29313.120
Microsoft.VisualStudio.Component.VC.MFC.ARM.Spectre	C++ MFC for latest v142 build tools with Spectre Mitigations (ARM)	16.4.29429.68
Microsoft.VisualStudio.Component.VC.MFC.ARM64	C++ MFC for latest v142 build tools (ARM64)	16.4.29313.120
Microsoft.VisualStudio.Component.VC.MFC.ARM64.Spectre	C++ MFC for latest v142 build tools with Spectre Mitigations (ARM64)	16.4.29429.68
Microsoft.VisualStudio.Component.VC.Redist.MSM	C++ 2019 Redistributable MSMs	16.4.29429.68
Microsoft.VisualStudio.Component.VC.Runtimes.ARM.Spectre	MSVC v142 - VS 2019 C++ ARM Spectre-mitigated libs (v14.24)	16.4.29429.68
Microsoft.VisualStudio.Component.VC.Runtimes.ARM64.Spectre	MSVC v142 - VS 2019 C++ ARM64 Spectre-mitigated libs (v14.24)	16.4.29429.68
Microsoft.VisualStudio.Component.VC.Runtimes.x86.x64.Spectre	MSVC v142 - VS 2019 C++ x64/x86 Spectre-mitigated libs (v14.24)	16.4.29429.68
Microsoft.VisualStudio.Component.VC.v141.ARM.Spectre	MSVC v141 - VS 2017 C++ ARM Spectre-mitigated libs (v14.16)	16.4.29429.68
Microsoft.VisualStudio.Component.VC.v141.ARM64.Spectre	MSVC v141 - VS 2017 C++ ARM64 Spectre-mitigated libs (v14.16)	16.4.29429.68
Microsoft.VisualStudio.Component.VC.v141.ATL	C++ ATL for v141 build tools (x86 & x64)	16.0.28625.61
Microsoft.VisualStudio.Component.VC.v141.ATL.ARM	C++ ATL for v141 build tools (ARM)	16.0.28625.61
Microsoft.VisualStudio.Component.VC.v141.ATL.ARM.Spectre	C++ ATL for v141 build tools with Spectre Mitigations (ARM)	16.0.28625.61

Component ID	Name	Version
Microsoft.VisualStudio.Component.VC.v141.ATL.ARM64	C++ ATL for v141 build tools (ARM64)	16.0.28625.61
Microsoft.VisualStudio.Component.VC.v141.ATL.ARM64.Spectre	C++ ATL for v141 build tools with Spectre Mitigations (ARM64)	16.0.28625.61
Microsoft.VisualStudio.Component.VC.v141.ATL.Spectre	C++ ATL for v141 build tools with Spectre Mitigations (x86 & x64)	16.0.28625.61
Microsoft.VisualStudio.Component.VC.v141.CLI.Support	C++/CLI support for v141 build tools (14.16)	16.3.29207.166
Microsoft.VisualStudio.Component.VC.v141.MFC	C++ MFC for v141 build tools (x86 & x64)	16.0.28625.61
Microsoft.VisualStudio.Component.VC.v141.MFC.ARM	C++ MFC for v141 build tools (ARM)	16.2.28915.88
Microsoft.VisualStudio.Component.VC.v141.MFC.ARM.Spectre	C++ MFC for v141 build tools with Spectre Mitigations (ARM)	16.0.28625.61
Microsoft.VisualStudio.Component.VC.v141.MFC.ARM64	C++ MFC for v141 build tools (ARM64)	16.0.28625.61
Microsoft.VisualStudio.Component.VC.v141.MFC.ARM64.Spectre	C++ MFC for v141 build tools with Spectre Mitigations (ARM64)	16.0.28625.61
Microsoft.VisualStudio.Component.VC.v141.MFC.Spectre	C++ MFC for v141 build tools with Spectre Mitigations (x86 & x64)	16.0.28625.61
Microsoft.VisualStudio.Component.VC.v141.x86.x64.Spectre	MSVC v141 - VS 2017 C++ x64/x86 Spectre-mitigated libs (v14.16)	16.4.29429.68
Microsoft.VisualStudio.Component.VisualStudioData	Data sources and service references	16.0.28707.177
Microsoft.VisualStudio.Component.WinXP	C++ Windows XP Support for VS 2017 (v141) tools [Deprecated]	16.1.28811.260
Microsoft.VisualStudio.Web.Mvc4.ComponentGroup	ASP.NET MVC 4	16.1.28810.153

Get support

Sometimes, things can go wrong. If your Visual Studio installation fails, see [Troubleshoot Visual Studio installation and upgrade issues](#) for step-by-step guidance.

We also offer a [live chat](#) (English only) support option for installation-related issues.

Here are a few more support options:

- Report product issues to us via the [Report a Problem](#) tool that appears both in the Visual Studio Installer and in the Visual Studio IDE.
- Suggest a feature, track product issues, and find answers in the [Visual Studio Developer Community](#).

- Use your [GitHub](#) account to talk to us and other Visual Studio developers in the [Visual Studio conversation in the Gitter community](#).

See also

- [Visual Studio workload and component IDs](#)
- [Visual Studio administrator guide](#)
- [Use command-line parameters to install Visual Studio](#)
 - [Command-line parameter examples](#)
- [Create an offline installation of Visual Studio](#)

Visual Studio Team Explorer component directory

12/4/2019 • 3 minutes to read • [Edit Online](#)

The tables on this page list the IDs that you can use to install Visual Studio from the command line, or that you can specify as a dependency in a VSIX manifest. We'll add additional components when we release updates to Visual Studio.

Also note the following about the page:

- Each workload has its own section, followed by the workload ID and a table of the components that are available for the workload.
- By default, the **Required** components will be installed when you install the workload.
- If you choose to, you can also install the **Recommended** and **Optional** components.
- We've also added a section that lists the additional components that are not affiliated with any workload.

When you set dependencies in your VSIX manifest, you must specify Component IDs only. Use the tables on this page to determine the minimum component dependencies. In some scenarios, this might mean that you specify only one component from a workload. In other scenarios, it might mean that you specify multiple components from a single workload or multiple components from multiple workloads. For more information, see the [How to: Migrate extensibility projects to Visual Studio 2017](#) page.

For more information about how to use these IDs, see the [Use command-line parameters to install Visual Studio 2017](#) page. And, for a list of workload and component IDs for other products, see the [Visual Studio workload and component IDs](#) page.

Visual Studio core editor (included with Visual Studio Team Explorer 2017)

ID: Microsoft.VisualStudio.Workload.CoreEditor

Description: The Visual Studio core shell experience, including syntax-aware code editing, source code control and work item management.

Components included by this workload

COMPONENT ID	NAME	VERSION	DEPENDENCY TYPE
Microsoft.VisualStudio.Component.CoreEditor	Visual Studio core editor	15.8.27729.1	Required
Microsoft.VisualStudio.Component.StartPageExperiment.Cpp	Visual Studio Start Page for C++ Users	15.0.27128.1	Optional

Unaffiliated components

These are components that are not included with any workload, but may be selected as an individual component.

COMPONENT ID	NAME	VERSION
n/a	n/a	n/a

When you set dependencies in your VSIX manifest, you must specify Component IDs only. Use the tables on this page to determine the minimum component dependencies. In some scenarios, this might mean that you specify only one component from a workload. In other scenarios, it might mean that you specify multiple components from a single workload or multiple components from multiple workloads. For more information, see the [How to: Migrate extensibility projects to Visual Studio](#) page.

For more information about how to use these IDs, see the [Use command-line parameters to install Visual Studio](#) page. And, for a list of workload and component IDs for other products, see the [Visual Studio workload and component IDs](#) page.

Visual Studio core editor (included with Visual Studio Team Explorer 2019)

ID: Microsoft.VisualStudio.Workload.CoreEditor

Description: The Visual Studio core shell experience, including syntax-aware code editing, source code control and work item management.

Components included by this workload

COMPONENT ID	NAME	VERSION	DEPENDENCY TYPE
Microsoft.VisualStudio.Component.CoreEditor	Visual Studio core editor	16.1.28811.260	Required
Microsoft.VisualStudio.Component.StartPageExperiment.Cpp	Visual Studio Start Page for C++ Users	16.0.28315.86	Optional

Unaffiliated components

These are components that are not included with any workload, but may be selected as an individual component.

COMPONENT ID	NAME	VERSION
n/a	n/a	n/a

Get support

Sometimes, things can go wrong. If your Visual Studio installation fails, see [Troubleshoot Visual Studio installation and upgrade issues](#) for step-by-step guidance.

We also offer a [Live chat](#) (English only) support option for installation-related issues.

Here are a few more support options:

- Report product issues to us via the [Report a Problem](#) tool that appears both in the Visual Studio Installer and in the Visual Studio IDE.
- Suggest a feature, track product issues, and find answers in the [Visual Studio Developer Community](#).
- Use your [GitHub](#) account to talk to us and other Visual Studio developers in the [Visual Studio conversation in the Gitter community](#).

See also

- [Visual Studio workload and component IDs](#)

- [Visual Studio administrator guide](#)
- [Use command-line parameters to install Visual Studio](#)
 - [Command-line parameter examples](#)
- [Create an offline installation of Visual Studio](#)

Visual Studio Desktop Express component directory

3/4/2019 • 3 minutes to read • [Edit Online](#)

The tables on this page list the IDs that you can use to install Visual Studio by using the command line or that you can specify as a dependency in a VSIX manifest. Note that we will add additional components as we release updates to Visual Studio.

Also note the following about the page:

- Each workload has its own section, followed by the workload ID and a table of the components that are available for the workload.
- By default, the **Required** components will be installed when you install the workload.
- If you choose to, you can also install the **Recommended** and **Optional** components.
- We've also added a section that lists the additional components that are not affiliated with any workload.

When you set dependencies in your VSIX manifest, you must specify Component IDs only. Use the tables on this page to determine our minimum component dependencies. In some scenarios, this might mean that you specify only one component from a workload. In other scenarios, it might mean that you specify multiple components from a single workload or multiple components from multiple workloads. For more information, see the [How to: Migrate Extensibility Projects to Visual Studio 2017](#) page.

For more information about how to use these IDs, see [Use Command-Line Parameters to Install Visual Studio 2017](#) page. And, for a list of workload and component IDs for other products, see [Visual Studio 2017 Workload and Component IDs](#) page.

Express for Windows Desktop

ID: Microsoft.VisualStudio.Workload.WDExpress

Description: Build Native and Managed applications like WPF, WinForms and Win32 with syntax-aware code editing, source code control, and work item management. Includes support for C#, Visual Basic, and Visual C++.

Components included by this workload

COMPONENT ID	NAME	VERSION	DEPENDENCY TYPE
Microsoft.Component.ClickOnce	ClickOnce Publishing	15.8.27825.0	Required
Microsoft.Component.HelpViewer	Help Viewer	15.6.27323.2	Required
Microsoft.Component_MSBUILD	MSBuild	15.7.27520.0	Required
Microsoft.Component.VC.Runtime.OSSupport	Visual C++ runtime for UWP	15.6.27406.0	Required
Microsoft.Net.Component.4.5.1.TargetingPack	.NET Framework 4.5.1 targeting pack	15.6.27406.0	Required

Component ID	Name	Version	Dependency Type
Microsoft.Net.Component.4.5.2.TargetingPack	.NET Framework 4.5.2 targeting pack	15.6.27406.0	Required
Microsoft.Net.Component.4.5.TargetingPack	.NET Framework 4.5 targeting pack	15.6.27406.0	Required
Microsoft.Net.Component.4.6.1.SDK	.NET Framework 4.6.1 SDK	15.6.27406.0	Required
Microsoft.Net.Component.4.6.1.TargetingPack	.NET Framework 4.6.1 targeting pack	15.6.27406.0	Required
Microsoft.Net.Component.4.6.TargetingPack	.NET Framework 4.6 targeting pack	15.6.27406.0	Required
Microsoft.Net.Component.4.TargetingPack	.NET Framework 4 targeting pack	15.6.27406.0	Required
Microsoft.Net.ComponentGroup.DevelopmentPrerequisites	.NET Framework 4.6.1 development tools	15.8.27825.0	Required
Microsoft.Net.ComponentGroup.TargetingPacks.Common	.NET Framework 4 – 4.6 development tools	15.6.27406.0	Required
Microsoft.VisualStudio.Component.Common.Azure.Tools	Connectivity and publishing tools	15.9.28107.0	Required
Microsoft.VisualStudio.Component.CoreEditor	Visual Studio core editor	15.8.27729.1	Required
Microsoft.VisualStudio.Component.EntityFramework	Entity Framework 6 tools	15.6.27406.0	Required
Microsoft.VisualStudio.Component.NuGet	NuGet package manager	15.9.28016.0	Required
Microsoft.VisualStudio.Component.Roslyn.Compiler	C# and Visual Basic Roslyn compilers	15.6.27309.0	Required
Microsoft.VisualStudio.Component.Roslyn.LanguageServices	C# and Visual Basic	15.8.27729.1	Required
Microsoft.VisualStudio.Component.SQL.ADAL	SQL ADAL runtime	15.6.27406.0	Required
Microsoft.VisualStudio.Component.SQL.CLR	CLR data types for SQL Server	15.0.26208.0	Required
Microsoft.VisualStudio.Component.SQL.CMDUtils	SQL Server Command Line Utilities	15.0.26208.0	Required

Component ID	Name	Version	Dependency Type
Microsoft.VisualStudio.Component.SQL.DataSources	Data sources for SQL Server support	15.0.26621.2	Required
Microsoft.VisualStudio.Component.SQL.LocalDB.Runtime	SQL Server Express 2016 LocalDB	15.7.27617.1	Required
Microsoft.VisualStudio.Component.SQL.NCLI	SQL Server Native Client	15.0.26208.0	Required
Microsoft.VisualStudio.Component.SQL.SSDT	SQL Server Data Tools	15.9.28107.0	Required
Microsoft.VisualStudio.Component.Static.Analysis.Tools	Static analysis tools	15.0.26208.0	Required
Microsoft.VisualStudio.Component.TextTemplating	Text Template Transformation	15.0.26208.0	Required
Microsoft.VisualStudio.Component.VC.CLI.Support	C++/CLI support	15.6.27309.0	Required
Microsoft.VisualStudio.Component.VC.Tools.ARM	Visual C++ compilers and libraries for ARM	15.8.27825.0	Required
Microsoft.VisualStudio.Component.VC.Tools.ARM64	Visual C++ compilers and libraries for ARM64	15.9.28230.55	Required
Microsoft.VisualStudio.Component.VisualStudioData	Data sources and service references	15.6.27406.0	Required
Microsoft.VisualStudio.Component.Windows10SDK.14393	Windows 10 SDK (10.0.14393.0)	15.6.27406.0	Required
Microsoft.VisualStudio.Component.Windows10SDK.17134	Windows 10 SDK (10.0.17134.0)	15.8.27924.0	Required

Unaffiliated components

These are components that are not included with any workload, but may be selected as an individual component.

Component ID	Name	Version
n/a	n/a	n/a

Get support

Sometimes, things can go wrong. If your Visual Studio installation fails, see [Troubleshoot Visual Studio installation and upgrade issues](#) for step-by-step guidance.

We also offer a [Live chat](#) (English only) support option for installation-related issues.

Here are a few more support options:

- Report product issues to us via the [Report a Problem](#) tool that appears both in the Visual Studio Installer and in the Visual Studio IDE.
- Suggest a feature, track product issues, and find answers in the [Visual Studio Developer Community](#).
- Use your [GitHub](#) account to talk to us and other Visual Studio developers in the [Visual Studio conversation in the Gitter community](#).

See also

- [Visual Studio workload and component IDs](#)
- [Visual Studio administrator guide](#)
- [Use command-line parameters to install Visual Studio](#)
 - [Command-line parameter examples](#)
- [Create an offline installation of Visual Studio](#)

Visual Studio Build Tools component directory

12/4/2019 • 31 minutes to read • [Edit Online](#)

The tables on this page list the IDs that you can use to install Visual Studio from the command line, or that you can specify as a dependency in a VSIX manifest. We'll add additional components when we release updates to Visual Studio.

Also note the following about the page:

- Each workload has its own section, followed by the workload ID and a table of the components that are available for the workload.
- By default, the **Required** components will be installed when you install the workload.
- If you choose to, you can also install the **Recommended** and **Optional** components.
- We've also added a section that lists the additional components that are not affiliated with any workload.

When you set dependencies in your VSIX manifest, you must specify Component IDs only. Use the tables on this page to determine the minimum component dependencies. In some scenarios, this might mean that you specify only one component from a workload. In other scenarios, it might mean that you specify multiple components from a single workload or multiple components from multiple workloads. For more information, see the [How to: Migrate extensibility projects to Visual Studio 2017](#) page.

For more information about how to use these IDs, see the [Use command-line parameters to install Visual Studio 2017](#) page. And, for a list of workload and component IDs for other products, see the [Visual Studio workload and component IDs](#) page.

To install these components using the manual installer, download and run the [Build Tools for Visual Studio](#).

Azure development build tools

ID: Microsoft.VisualStudio.Workload.AzureBuildTools

Description: MSBuild tasks and targets for building Azure applications.

Components included by this workload

COMPONENT ID	NAME	VERSION	DEPENDENCY TYPE
Microsoft.Net.Component.4.6.1.SDK	.NET Framework 4.6.1 SDK	15.6.27406.0	Required
Microsoft.Net.Component.4.6.1.TargetingPack	.NET Framework 4.6.1 targeting pack	15.6.27406.0	Required
Microsoft.Net.ComponentGroup.DevelopmentPrerequisites	.NET Framework 4.6.1 development tools	15.8.27825.0	Required
Microsoft.VisualStudio.Component.Azure.AuthoringTools	Azure Authoring Tools	15.9.28307.421	Required
Microsoft.VisualStudio.Component.Azure.ClientLibs	Azure libraries for .NET	15.0.26208.0	Required

Component ID	Name	Version	Dependency Type
Microsoft.VisualStudio.Component.Azure.Waverton.BuildTools	Azure Cloud Services build tools	15.7.27617.1	Required
Microsoft.VisualStudio.Component.DockerTools.BuildTools	Container development tools - Build Tools	15.7.27617.1	Required
Microsoft.VisualStudio.Component.NuGet.BuildTools	NuGet targets and build tasks	15.9.28016.0	Required
Microsoft.VisualStudio.Component.TypeScript.3.1	TypeScript 3.1 SDK	15.0.28218.60	Required
Microsoft.VisualStudio.Wcf.BuildTools.ComponentGroup	WCF development build tools	15.6.27309.0	Required
Microsoft.VisualStudio.Web.BuildTools.ComponentGroup	Web development build tools	15.8.27729.1	Required
Microsoft.Net.Component.4.5.1.TargetingPack	.NET Framework 4.5.1 targeting pack	15.6.27406.0	Recommended
Microsoft.Net.Component.4.5.2.TargetingPack	.NET Framework 4.5.2 targeting pack	15.6.27406.0	Recommended
Microsoft.Net.Component.4.5.TargetingPack	.NET Framework 4.5 targeting pack	15.6.27406.0	Recommended
Microsoft.Net.Component.4.6.TargetingPack	.NET Framework 4.6 targeting pack	15.6.27406.0	Recommended
Microsoft.Net.Component.4.TargetingPack	.NET Framework 4 targeting pack	15.6.27406.0	Recommended
Microsoft.Net.ComponentGroup.TargetingPacks.Common	.NET Framework 4 – 4.6 development tools	15.6.27406.0	Recommended
Microsoft.Net.Core.Component.SDK.2.1	.NET Core 2.1 development tools	15.8.27924.0	Recommended
Microsoft.VisualStudio.Component.AspNet45	Advanced ASP.NET features	15.7.27625.0	Recommended
Microsoft.VisualStudio.Component.WebDeploy	Web Deploy	15.8.27729.1	Recommended
Microsoft.Net.Component.3.5.DeveloperTools	.NET Framework 3.5 development tools	15.6.27406.0	Optional
Microsoft.Net.Component.4.6.2.SDK	.NET Framework 4.6.2 SDK	15.6.27406.0	Optional

Component ID	Name	Version	Dependency Type
Microsoft.Net.Component.4.6.2.TargetingPack	.NET Framework 4.6.2 targeting pack	15.6.27406.0	Optional
Microsoft.Net.Component.4.7.1.SDK	.NET Framework 4.7.1 SDK	15.6.27406.0	Optional
Microsoft.Net.Component.4.7.1.TargetingPack	.NET Framework 4.7.1 targeting pack	15.6.27406.0	Optional
Microsoft.Net.Component.4.7.2.SDK	.NET Framework 4.7.2 SDK	15.8.27825.0	Optional
Microsoft.Net.Component.4.7.2.TargetingPack	.NET Framework 4.7.2 targeting pack	15.8.27825.0	Optional
Microsoft.Net.Component.4.7.SDK	.NET Framework 4.7 SDK	15.6.27406.0	Optional
Microsoft.Net.Component.4.7.TargetingPack	.NET Framework 4.7 targeting pack	15.6.27406.0	Optional
Microsoft.Net.ComponentGroup.4.6.2.DeveloperTools	.NET Framework 4.6.2 development tools	15.6.27406.0	Optional
Microsoft.Net.ComponentGroup.4.7.1.DeveloperTools	.NET Framework 4.7.1 development tools	15.6.27406.0	Optional
Microsoft.Net.ComponentGroup.4.7.2.DeveloperTools	.NET Framework 4.7.2 development tools	15.8.27825.0	Optional
Microsoft.Net.ComponentGroup.4.7.DeveloperTools	.NET Framework 4.7 development tools	15.6.27406.0	Optional

Data storage and processing build tools

ID: Microsoft.VisualStudio.Workload.DataBuildTools

Description: Build SQL Server Database Projects

Components included by this workload

Component ID	Name	Version	Dependency Type
Microsoft.Net.Component.4.5.1.TargetingPack	.NET Framework 4.5.1 targeting pack	15.6.27406.0	Recommended
Microsoft.Net.Component.4.5.2.TargetingPack	.NET Framework 4.5.2 targeting pack	15.6.27406.0	Recommended
Microsoft.Net.Component.4.5.TargetingPack	.NET Framework 4.5 targeting pack	15.6.27406.0	Recommended
Microsoft.Net.Component.4.6.TargetingPack	.NET Framework 4.6 targeting pack	15.6.27406.0	Recommended

Component ID	Name	Version	Dependency Type
Microsoft.Net.Component.4.TargetingPack	.NET Framework 4 targeting pack	15.6.27406.0	Recommended
Microsoft.Net.ComponentGroup.TargetingPacks.Common	.NET Framework 4 – 4.6 development tools	15.6.27406.0	Recommended
Microsoft.VisualStudio.Component.Roslyn.Compiler	C# and Visual Basic Roslyn compilers	15.6.27309.0	Recommended
Microsoft.VisualStudio.Component.Roslyn.LanguageServices	C# and Visual Basic	15.8.27729.1	Recommended
Microsoft.VisualStudio.Component.SQL.SSDTBuildSku	SQL Server Data Tools - Build Tools	15.8.27825.0	Recommended
Microsoft.VisualStudio.Component.Static.Analysis.Tools	Static analysis tools	15.0.26208.0	Recommended

.NET desktop build tools

ID: Microsoft.VisualStudio.Workload.ManagedDesktopBuildTools

Description: Tools for building WPF, Windows Forms, and console applications using C#, Visual Basic, and F#.

Components included by this workload

Component ID	Name	Version	Dependency Type
Microsoft.Component.MSBuild	MSBuild	15.7.27520.0	Required
Microsoft.Net.Component.4.6.1.SDK	.NET Framework 4.6.1 SDK	15.6.27406.0	Required
Microsoft.Net.Component.4.6.1.TargetingPack	.NET Framework 4.6.1 targeting pack	15.6.27406.0	Required
Microsoft.VisualStudio.Component.NuGet.BuildTools	NuGet targets and build tasks	15.9.28016.0	Required
Microsoft.VisualStudio.Component.Roslyn.Compiler	C# and Visual Basic Roslyn compilers	15.6.27309.0	Required
Microsoft.Component.ClickOnce.MSBuild	ClickOnce Build Tools	15.7.27617.1	Recommended
Microsoft.Net.Component.4.5.1.TargetingPack	.NET Framework 4.5.1 targeting pack	15.6.27406.0	Recommended
Microsoft.Net.Component.4.5.2.TargetingPack	.NET Framework 4.5.2 targeting pack	15.6.27406.0	Recommended

Component ID	Name	Version	Dependency Type
Microsoft.Net.Component.4.5.TargetingPack	.NET Framework 4.5 targeting pack	15.6.27406.0	Recommended
Microsoft.Net.Component.4.6.TargetingPack	.NET Framework 4.6 targeting pack	15.6.27406.0	Recommended
Microsoft.Net.Component.4.TargetingPack	.NET Framework 4 targeting pack	15.6.27406.0	Recommended
Microsoft.Net.ComponentGroup.TargetingPacks.Common	.NET Framework 4 – 4.6 development tools	15.6.27406.0	Recommended
Microsoft.Net.Core.Component.SDK	.NET Core 2.0 development tools	15.6.27406.0	Recommended
Microsoft.Net.Core.Component.SDK.2.1	.NET Core 2.1 development tools	15.8.27924.0	Recommended
Microsoft.VisualStudio.Component.TestTools.BuildTools	Testing tools core features - Build Tools	15.7.27625.0	Recommended
Microsoft.VisualStudio.Wcf.BuildTools.ComponentGroup	WCF development build tools	15.6.27309.0	Recommended
Microsoft.Net.Component.3.5.DeveloperTools	.NET Framework 3.5 development tools	15.6.27406.0	Optional
Microsoft.Net.Component.4.6.2.SDK	.NET Framework 4.6.2 SDK	15.6.27406.0	Optional
Microsoft.Net.Component.4.6.2.TargetingPack	.NET Framework 4.6.2 targeting pack	15.6.27406.0	Optional
Microsoft.Net.Component.4.7.1.SDK	.NET Framework 4.7.1 SDK	15.6.27406.0	Optional
Microsoft.Net.Component.4.7.1.TargetingPack	.NET Framework 4.7.1 targeting pack	15.6.27406.0	Optional
Microsoft.Net.Component.4.7.2.SDK	.NET Framework 4.7.2 SDK	15.8.27825.0	Optional
Microsoft.Net.Component.4.7.2.TargetingPack	.NET Framework 4.7.2 targeting pack	15.8.27825.0	Optional
Microsoft.Net.Component.4.7.SDK	.NET Framework 4.7 SDK	15.6.27406.0	Optional
Microsoft.Net.Component.4.7.TargetingPack	.NET Framework 4.7 targeting pack	15.6.27406.0	Optional
Microsoft.Net.ComponentGroup.4.6.2.DeveloperTools	.NET Framework 4.6.2 development tools	15.6.27406.0	Optional

Component ID	Name	Version	Dependency Type
Microsoft.Net.ComponentGroup.4.7.1.DeveloperTools	.NET Framework 4.7.1 development tools	15.6.27406.0	Optional
Microsoft.Net.ComponentGroup.4.7.2.DeveloperTools	.NET Framework 4.7.2 development tools	15.8.27825.0	Optional
Microsoft.Net.ComponentGroup.4.7.DeveloperTools	.NET Framework 4.7 development tools	15.6.27406.0	Optional
Microsoft.Net.Core.Component.SDK.1x	.NET Core 1.0 - 1.1 development tools	15.6.27406.0	Optional
Microsoft.VisualStudio.Component.FSharp.MSBuild	F# compiler	15.8.27825.0	Optional

MSBuild Tools

ID: Microsoft.VisualStudio.Workload.MSBuildTools

Description: Provides the tools required to build MSBuild-based applications.

Components included by this workload

Component ID	Name	Version	Dependency Type
Microsoft.Component.MSBuild	MSBuild	15.7.27520.0	Required
Microsoft.VisualStudio.Component.CoreBuildTools	Visual Studio Build Tools Core	15.6.27309.0	Required
Microsoft.VisualStudio.Component.Roslyn.Compiler	C# and Visual Basic Roslyn compilers	15.6.27309.0	Required

.NET Core build tools

ID: Microsoft.VisualStudio.Workload.NetCoreBuildTools

Description: Tools for building applications using .NET Core, ASP.NET Core, HTML/JavaScript, and Containers.

Components included by this workload

Component ID	Name	Version	Dependency Type
Microsoft.Net.Core.Component.SDK.2.1	.NET Core 2.1 development tools	15.8.27924.0	Required
Microsoft.NetCore.BuildTools.ComponentGroup	.NET Core build tools	15.8.27906.1	Required
Microsoft.VisualStudio.Component.NuGet.BuildTools	NuGet targets and build tasks	15.9.28016.0	Required

COMPONENT ID	NAME	VERSION	DEPENDENCY TYPE
Microsoft.Net.Core.Component.SDK	.NET Core 2.0 development tools	15.6.27406.0	Optional
Microsoft.Net.Core.Component.SDK.1x	.NET Core 1.0 - 1.1 development tools	15.6.27406.0	Optional

Node.js build tools

ID: Microsoft.VisualStudio.Workload.NodeBuildTools

Description: MSBuild tasks and targets for building scalable network applications using Node.js, an asynchronous event-driven JavaScript runtime.

Components included by this workload

COMPONENT ID	NAME	VERSION	DEPENDENCY TYPE
Microsoft.VisualStudio.Component.Node.Build	Node.js MSBuild support	15.8.27825.0	Required
Microsoft.VisualStudio.Component.TypeScript.3.1	TypeScript 3.1 SDK	15.0.28218.60	Required

Office/SharePoint build tools

ID: Microsoft.VisualStudio.Workload.OfficeBuildTools

Description: Build Office and SharePoint add-ins, and VSTO add-ins.

Components included by this workload

COMPONENT ID	NAME	VERSION	DEPENDENCY TYPE
Microsoft.Component.ClickOnce.MSBuild	ClickOnce Build Tools	15.7.27617.1	Required
Microsoft.Component.MSBuild	MSBuild	15.7.27520.0	Required
Microsoft.Net.Component.4.5.2.TargetingPack	.NET Framework 4.5.2 targeting pack	15.6.27406.0	Required
Microsoft.Net.Component.4.5.TargetingPack	.NET Framework 4.5 targeting pack	15.6.27406.0	Required
Microsoft.Net.Component.4.6.1.SDK	.NET Framework 4.6.1 SDK	15.6.27406.0	Required
Microsoft.Net.Component.4.6.1.TargetingPack	.NET Framework 4.6.1 targeting pack	15.6.27406.0	Required
Microsoft.Net.Component.4.TargetingPack	.NET Framework 4 targeting pack	15.6.27406.0	Required

Component ID	Name	Version	Dependency Type
Microsoft.Net.ComponentGroup.DevelopmentPrerequisites	.NET Framework 4.6.1 development tools	15.8.27825.0	Required
Microsoft.VisualStudio.Component.NuGet	NuGet package manager	15.9.28016.0	Required
Microsoft.VisualStudio.Component.NuGet.BuildTools	NuGet targets and build tasks	15.9.28016.0	Required
Microsoft.VisualStudio.Component.Roslyn.Compiler	C# and Visual Basic Roslyn compilers	15.6.27309.0	Required
Microsoft.VisualStudio.Component.Sharepoint.BuildTools	Office/SharePoint development build tools	15.8.27825.0	Required
Microsoft.VisualStudio.Component.Workflow.BuildTools	Windows Workflow Foundation Build Tools	15.8.27906.1	Required
Microsoft.VisualStudio.Wcf.BuildTools.ComponentGroup	WCF development build tools	15.6.27309.0	Required
Microsoft.VisualStudio.Web.BuildTools.ComponentGroup	Web development build tools	15.8.27729.1	Required
Microsoft.VisualStudio.Component.TeamOffice.BuildTools	Visual Studio Tools for Office (VSTO) build tools	15.7.27617.1	Recommended
Microsoft.VisualStudio.Component.WebDeploy	Web Deploy	15.8.27729.1	Recommended
Microsoft.Net.Component.4.6.2.SDK	.NET Framework 4.6.2 SDK	15.6.27406.0	Optional
Microsoft.Net.Component.4.6.2.TargetingPack	.NET Framework 4.6.2 targeting pack	15.6.27406.0	Optional
Microsoft.Net.Component.4.7.1.SDK	.NET Framework 4.7.1 SDK	15.6.27406.0	Optional
Microsoft.Net.Component.4.7.1.TargetingPack	.NET Framework 4.7.1 targeting pack	15.6.27406.0	Optional
Microsoft.Net.Component.4.7.2.SDK	.NET Framework 4.7.2 SDK	15.8.27825.0	Optional
Microsoft.Net.Component.4.7.2.TargetingPack	.NET Framework 4.7.2 targeting pack	15.8.27825.0	Optional
Microsoft.Net.Component.4.7.SDK	.NET Framework 4.7 SDK	15.6.27406.0	Optional

Component ID	Name	Version	Dependency Type
Microsoft.Net.Component.4.7.TargetingPack	.NET Framework 4.7 targeting pack	15.6.27406.0	Optional
Microsoft.Net.ComponentGroup.4.6.2.DeveloperTools	.NET Framework 4.6.2 development tools	15.6.27406.0	Optional
Microsoft.Net.ComponentGroup.4.7.1.DeveloperTools	.NET Framework 4.7.1 development tools	15.6.27406.0	Optional
Microsoft.Net.ComponentGroup.4.7.2.DeveloperTools	.NET Framework 4.7.2 development tools	15.8.27825.0	Optional
Microsoft.Net.ComponentGroup.4.7.DeveloperTools	.NET Framework 4.7 development tools	15.6.27406.0	Optional

Universal Windows Platform build tools

ID: Microsoft.VisualStudio.Workload.UniversalBuildTools

Description: Provides the tools required to build Universal Windows Platform applications.

Components included by this workload

Component ID	Name	Version	Dependency Type
Microsoft.Component.MSBuild	MSBuild	15.7.27520.0	Required
Microsoft.Component.NetFX.Native	.NET Native	15.0.26208.0	Required
Microsoft.Component.VC.Runtime.OSSupport	Visual C++ runtime for UWP	15.6.27406.0	Required
Microsoft.Net.Component.4.7.1.SDK	.NET Framework 4.7.1 SDK	15.6.27406.0	Required
Microsoft.VisualStudio.Component.NuGet.BuildTools	NuGet targets and build tasks	15.9.28016.0	Required
Microsoft.VisualStudio.Component.Roslyn.Compiler	C# and Visual Basic Roslyn compilers	15.6.27309.0	Required
Microsoft.VisualStudio.Component.Static.Analysis.Tools	Static analysis tools	15.0.26208.0	Required
Microsoft.VisualStudio.Component.VC.Tools.ARM	Visual C++ compilers and libraries for ARM	15.8.27825.0	Required
Microsoft.VisualStudio.Component.VC.Tools.x86.x64	VC++ 2017 version 15.9 v14.16 latest v141 tools	15.9.28230.55	Required

Component ID	Name	Version	Dependency Type
Microsoft.VisualStudio.ComponentGroup.UWP.BuildTools	Universal Windows Platform build prerequisites	15.8.27705.0	Required
Microsoft.VisualStudio.Component.Windows10SDK.17763	Windows 10 SDK (10.0.17763.0)	15.9.28307.102	Recommended
Microsoft.VisualStudio.Component.Windows10SDK.10240	Windows 10 SDK (10.0.10240.0)	15.6.27406.0	Optional
Microsoft.VisualStudio.Component.Windows10SDK.10586	Windows 10 SDK (10.0.10586.0)	15.6.27406.0	Optional
Microsoft.VisualStudio.Component.Windows10SDK.14393	Windows 10 SDK (10.0.14393.0)	15.6.27406.0	Optional
Microsoft.VisualStudio.Component.Windows10SDK.15063.Desktop	Windows 10 SDK (10.0.15063.0) for Desktop C++ [x86 and x64]	15.6.27406.0	Optional
Microsoft.VisualStudio.Component.Windows10SDK.15063.UWP	Windows 10 SDK (10.0.15063.0) for UWP: C#, VB, JS	15.6.27406.0	Optional
Microsoft.VisualStudio.Component.Windows10SDK.15063.UWP.Native	Windows 10 SDK (10.0.15063.0) for UWP: C++	15.6.27406.0	Optional
Microsoft.VisualStudio.Component.Windows10SDK.16299.Desktop	Windows 10 SDK (10.0.16299.0) for Desktop C++ [x86 and x64]	15.6.27406.0	Optional
Microsoft.VisualStudio.Component.Windows10SDK.16299.Desktop.arm	Windows 10 SDK (10.0.16299.0) for Desktop C++ [ARM and ARM64]	15.6.27406.0	Optional
Microsoft.VisualStudio.Component.Windows10SDK.16299.UWP	Windows 10 SDK (10.0.16299.0) for UWP: C#, VB, JS	15.6.27406.0	Optional
Microsoft.VisualStudio.Component.Windows10SDK.16299.UWP.Native	Windows 10 SDK (10.0.16299.0) for UWP: C++	15.6.27406.0	Optional
Microsoft.VisualStudio.Component.Windows10SDK.17134	Windows 10 SDK (10.0.17134.0)	15.9.28307.102	Optional
Microsoft.VisualStudio.ComponentGroup.Windows10SDK.15063	Windows 10 SDK (10.0.15063.0)	15.8.27825.0	Optional

Component ID	Name	Version	Dependency Type
Microsoft.VisualStudio.ComponentGroup.Windows10SDK.K.16299	Windows 10 SDK (10.0.16299.0)	15.8.27825.0	Optional

Visual C++ build tools

ID: Microsoft.VisualStudio.Workload.VCTools

Description: Build Windows desktop applications using the Microsoft C++ toolset, ATL, or MFC.

Components included by this workload

Component ID	Name	Version	Dependency Type
Microsoft.VisualStudio.Component.Static.Analysis.Tools	Static analysis tools	15.0.26208.0	Required
Microsoft.VisualStudio.Component.VC.CoreBuildTools	Visual C++ Build Tools core features	15.8.27729.1	Required
Microsoft.VisualStudio.Component.VC.Redist.14.Latest	Visual C++ 2017 Redistributable Update	15.6.27406.0	Required
Microsoft.VisualStudio.Component.VC.Tools.x86.x64	VC++ 2017 version 15.9 v14.16 latest v141 tools	15.9.28230.55	Required
Microsoft.VisualStudio.Component.Windows10SDK	Windows Universal C Runtime	15.6.27406.0	Required
Microsoft.VisualStudio.Component.TestTools.BuildTools	Testing tools core features - Build Tools	15.7.27625.0	Recommended
Microsoft.VisualStudio.Component.VC.CMake.Project	Visual C++ tools for CMake	15.9.28307.102	Recommended
Microsoft.VisualStudio.Component.Windows10SDK.17763	Windows 10 SDK (10.0.17763.0)	15.9.28307.102	Recommended
Microsoft.Component.VC.Runtime.UCRTSDK	Windows Universal CRT SDK	15.6.27309.0	Optional
Microsoft.Net.Component.4.6.1.SDK	.NET Framework 4.6.1 SDK	15.6.27406.0	Optional
Microsoft.Net.Component.4.6.1.TargetingPack	.NET Framework 4.6.1 targeting pack	15.6.27406.0	Optional
Microsoft.VisualStudio.Component.VC.140	VC++ 2015.3 v14.00 (v140) toolset for desktop	15.7.27617.1	Optional
Microsoft.VisualStudio.Component.VC.ATL	Visual C++ ATL for x86 and x64	15.7.27625.0	Optional

Component ID	Name	Version	Dependency Type
Microsoft.VisualStudio.Component.VC.ATLMFC	Visual C++ MFC for x86 and x64	15.7.27625.0	Optional
Microsoft.VisualStudio.Component.VC.CLI.Support	C++/CLI support	15.6.27309.0	Optional
Microsoft.VisualStudio.Component.VC.Modules.x86.x64	Modules for Standard Library (experimental)	15.6.27309.0	Optional
Microsoft.VisualStudio.Component.VC.Tools.ARM	Visual C++ compilers and libraries for ARM	15.8.27825.0	Optional
Microsoft.VisualStudio.Component.VC.Tools.ARM64	Visual C++ compilers and libraries for ARM64	15.9.28230.55	Optional
Microsoft.VisualStudio.Component.Windows10SDK.10240	Windows 10 SDK (10.0.10240.0)	15.6.27406.0	Optional
Microsoft.VisualStudio.Component.Windows10SDK.10586	Windows 10 SDK (10.0.10586.0)	15.6.27406.0	Optional
Microsoft.VisualStudio.Component.Windows10SDK.14393	Windows 10 SDK (10.0.14393.0)	15.6.27406.0	Optional
Microsoft.VisualStudio.Component.Windows10SDK.15063.Desktop	Windows 10 SDK (10.0.15063.0) for Desktop C++ [x86 and x64]	15.6.27406.0	Optional
Microsoft.VisualStudio.Component.Windows10SDK.15063.UWP	Windows 10 SDK (10.0.15063.0) for UWP: C#, VB, JS	15.6.27406.0	Optional
Microsoft.VisualStudio.Component.Windows10SDK.15063.UWP.Native	Windows 10 SDK (10.0.15063.0) for UWP: C++	15.6.27406.0	Optional
Microsoft.VisualStudio.Component.Windows10SDK.16299.Desktop	Windows 10 SDK (10.0.16299.0) for Desktop C++ [x86 and x64]	15.6.27406.0	Optional
Microsoft.VisualStudio.Component.Windows10SDK.16299.Desktop.arm	Windows 10 SDK (10.0.16299.0) for Desktop C++ [ARM and ARM64]	15.6.27406.0	Optional
Microsoft.VisualStudio.Component.Windows10SDK.16299.UWP	Windows 10 SDK (10.0.16299.0) for UWP: C#, VB, JS	15.6.27406.0	Optional
Microsoft.VisualStudio.Component.Windows10SDK.16299.UWP.Native	Windows 10 SDK (10.0.16299.0) for UWP: C++	15.6.27406.0	Optional

Component ID	Name	Version	Dependency Type
Microsoft.VisualStudio.Component.Windows10SDK.17134	Windows 10 SDK (10.0.17134.0)	15.9.28307.102	Optional
Microsoft.VisualStudio.Component.Windows81SDK	Windows 8.1 SDK	15.6.27406.0	Optional
Microsoft.VisualStudio.Component.WinXP	Windows XP support for C++	15.8.27924.0	Optional
Microsoft.VisualStudio.ComponentGroup.NativeDesktop.Win81	Windows 8.1 SDK and UCRT SDK	15.6.27406.0	Optional
Microsoft.VisualStudio.ComponentGroup.NativeDesktop.WinXP	Windows XP support for C++	15.8.27705.0	Optional
Microsoft.VisualStudio.ComponentGroup.Windows10SDK.15063	Windows 10 SDK (10.0.15063.0)	15.8.27825.0	Optional
Microsoft.VisualStudio.ComponentGroup.Windows10SDK.16299	Windows 10 SDK (10.0.16299.0)	15.8.27825.0	Optional

Visual Studio extension development

ID: Microsoft.VisualStudio.Workload.VisualStudioExtensionBuildTools

Description: Tools for building add-ons and extensions for Visual Studio, including new commands, code analyzers and tool windows.

Components included by this workload

Component ID	Name	Version	Dependency Type
Microsoft.Component.MSBuild	MSBuild	15.7.27520.0	Required
Microsoft.Net.Component.4.6.1.SDK	.NET Framework 4.6.1 SDK	15.6.27406.0	Required
Microsoft.Net.Component.4.6.1.TargetingPack	.NET Framework 4.6.1 targeting pack	15.6.27406.0	Required
Microsoft.Net.Component.4.6.TargetingPack	.NET Framework 4.6 targeting pack	15.6.27406.0	Required
Microsoft.Net.ComponentGroup.DevelopmentPrerequisites	.NET Framework 4.6.1 development tools	15.8.27825.0	Required

COMPONENT ID	NAME	VERSION	DEPENDENCY TYPE
Microsoft.VisualStudio.Component.NuGet.BuildTools	NuGet targets and build tasks	15.9.28016.0	Required
Microsoft.VisualStudio.Component.Roslyn.Compiler	C# and Visual Basic Roslyn compilers	15.6.27309.0	Required
Microsoft.VisualStudio.Component.VSSDKBuildTools	Visual Studio SDK Build Tools Core	15.8.27924.0	Required
Microsoft.VisualStudio.ComponentGroup.VisualStudioExtensionBuildTools.Prerequisites	Visual Studio extension development prerequisites	15.8.27729.1	Required
Component.Dotfuscator	PreEmptive Protection - Dotfuscator	15.0.26208.0	Optional
Microsoft.Component.VC.Runtime.OSSupport	Visual C++ runtime for UWP	15.6.27406.0	Optional
Microsoft.VisualStudio.Component.Static.Analysis.Tools	Static analysis tools	15.0.26208.0	Optional
Microsoft.VisualStudio.Component.VC.ATL	Visual C++ ATL for x86 and x64	15.7.27625.0	Optional
Microsoft.VisualStudio.Component.VC.ATLMFC	Visual C++ MFC for x86 and x64	15.7.27625.0	Optional
Microsoft.VisualStudio.Component.VC.Tools.x86.x64	VC++ 2017 version 15.9 v14.16 latest v141 tools	15.9.28230.55	Optional

Web development build tools

ID: Microsoft.VisualStudio.Workload.WebBuildTools

Description: MS Build tasks and targets for building web applications.

Components included by this workload

COMPONENT ID	NAME	VERSION	DEPENDENCY TYPE
Microsoft.Net.Component.4.6.1.SDK	.NET Framework 4.6.1 SDK	15.6.27406.0	Required
Microsoft.Net.Component.4.6.1.TargetingPack	.NET Framework 4.6.1 targeting pack	15.6.27406.0	Required
Microsoft.Net.ComponentGroup.DevelopmentPrerequisites	.NET Framework 4.6.1 development tools	15.8.27825.0	Required
Microsoft.VisualStudio.Component.NuGet.BuildTools	NuGet targets and build tasks	15.9.28016.0	Required

Component ID	Name	Version	Dependency Type
Microsoft.VisualStudio.Component.TypeScript.3.1	TypeScript 3.1 SDK	15.0.28218.60	Required
Microsoft.VisualStudio.Web.BuildTools.ComponentGroup	Web development build tools	15.8.27729.1	Required
Microsoft.Component.ClickOnce.MSBuild	ClickOnce Build Tools	15.7.27617.1	Recommended
Microsoft.Net.Component.4.5.1.TargetingPack	.NET Framework 4.5.1 targeting pack	15.6.27406.0	Recommended
Microsoft.Net.Component.4.5.2.TargetingPack	.NET Framework 4.5.2 targeting pack	15.6.27406.0	Recommended
Microsoft.Net.Component.4.5.TargetingPack	.NET Framework 4.5 targeting pack	15.6.27406.0	Recommended
Microsoft.Net.Component.4.6.TargetingPack	.NET Framework 4.6 targeting pack	15.6.27406.0	Recommended
Microsoft.Net.Component.4.TargetingPack	.NET Framework 4 targeting pack	15.6.27406.0	Recommended
Microsoft.Net.ComponentGroup.TargetingPacks.Common	.NET Framework 4 – 4.6 development tools	15.6.27406.0	Recommended
Microsoft.Net.Core.Component.SDK.2.1	.NET Core 2.1 development tools	15.8.27924.0	Recommended
Microsoft.VisualStudio.Component.AspNet45	Advanced ASP.NET features	15.7.27625.0	Recommended
Microsoft.VisualStudio.Component.DockerTools.BuildTools	Container development tools - Build Tools	15.7.27617.1	Recommended
Microsoft.VisualStudio.Component.TestTools.BuildTools	Testing tools core features - Build Tools	15.7.27625.0	Recommended
Microsoft.VisualStudio.Component.WebDeploy	Web Deploy	15.8.27729.1	Recommended
Microsoft.VisualStudio.Wcf.BuildTools.ComponentGroup	WCF development build tools	15.6.27309.0	Recommended
Microsoft.Net.Component.3.5.DeveloperTools	.NET Framework 3.5 development tools	15.6.27406.0	Optional
Microsoft.Net.Component.4.6.2.SDK	.NET Framework 4.6.2 SDK	15.6.27406.0	Optional

COMPONENT ID	NAME	VERSION	DEPENDENCY TYPE
Microsoft.Net.Component.4.6.2.TargetingPack	.NET Framework 4.6.2 targeting pack	15.6.27406.0	Optional
Microsoft.Net.Component.4.7.1.SDK	.NET Framework 4.7.1 SDK	15.6.27406.0	Optional
Microsoft.Net.Component.4.7.1.TargetingPack	.NET Framework 4.7.1 targeting pack	15.6.27406.0	Optional
Microsoft.Net.Component.4.7.2.SDK	.NET Framework 4.7.2 SDK	15.8.27825.0	Optional
Microsoft.Net.Component.4.7.2.TargetingPack	.NET Framework 4.7.2 targeting pack	15.8.27825.0	Optional
Microsoft.Net.Component.4.7.SDK	.NET Framework 4.7 SDK	15.6.27406.0	Optional
Microsoft.Net.Component.4.7.TargetingPack	.NET Framework 4.7 targeting pack	15.6.27406.0	Optional
Microsoft.Net.ComponentGroup.4.6.2.DeveloperTools	.NET Framework 4.6.2 development tools	15.6.27406.0	Optional
Microsoft.Net.ComponentGroup.4.7.1.DeveloperTools	.NET Framework 4.7.1 development tools	15.6.27406.0	Optional
Microsoft.Net.ComponentGroup.4.7.2.DeveloperTools	.NET Framework 4.7.2 development tools	15.8.27825.0	Optional
Microsoft.Net.ComponentGroup.4.7.DeveloperTools	.NET Framework 4.7 development tools	15.6.27406.0	Optional
Microsoft.Net.Core.Component.SDK	.NET Core 2.0 development tools	15.6.27406.0	Optional
Microsoft.Net.Core.Component.SDK.1x	.NET Core 1.0 - 1.1 development tools	15.6.27406.0	Optional

Mobile Development with .NET

ID: Microsoft.VisualStudio.Workload.XamarinBuildTools

Description: Tools for building cross-platform applications for iOS, Android and Windows using C# and F#.

Components included by this workload

COMPONENT ID	NAME	VERSION	DEPENDENCY TYPE
Microsoft.Component.MSBuild	MSBuild	15.7.27520.0	Required
Microsoft.Net.Component.4.6.1.SDK	.NET Framework 4.6.1 SDK	15.6.27406.0	Required

Component ID	Name	Version	Dependency Type
Microsoft.Net.Component.4.6.1.TargetingPack	.NET Framework 4.6.1 targeting pack	15.6.27406.0	Required
Microsoft.VisualStudio.Component.NuGet.BuildTools	NuGet targets and build tasks	15.9.28016.0	Required
Microsoft.VisualStudio.Component.Roslyn.Compiler	C# and Visual Basic Roslyn compilers	15.6.27309.0	Required
Component.Android.SDK25	Android SDK setup (API level 25)	15.9.28107.0	Optional
Component.OpenJDK	Microsoft distribution OpenJDK	15.9.28125.51	Optional

Unaffiliated components

These are components that are not included with any workload, but may be selected as an individual component.

Component ID	Name	Version
Microsoft.VisualStudio.Component.TypeScript.2.0	TypeScript 2.0 SDK	15.8.27729.1
Microsoft.VisualStudio.Component.TypeScript.2.1	TypeScript 2.1 SDK	15.8.27729.1
Microsoft.VisualStudio.Component.TypeScript.2.2	TypeScript 2.2 SDK	15.8.27729.1
Microsoft.VisualStudio.Component.TypeScript.2.3	TypeScript 2.3 SDK	15.8.27729.1
Microsoft.VisualStudio.Component.TypeScript.2.5	TypeScript 2.5 SDK	15.6.27406.0
Microsoft.VisualStudio.Component.TypeScript.2.6	TypeScript 2.6 SDK	15.0.27729.1
Microsoft.VisualStudio.Component.TypeScript.2.7	TypeScript 2.7 SDK	15.0.27729.1
Microsoft.VisualStudio.Component.TypeScript.2.8	TypeScript 2.8 SDK	15.0.27729.1
Microsoft.VisualStudio.Component.TypeScript.2.9	TypeScript 2.9 SDK	15.0.27924.0
Microsoft.VisualStudio.Component.TypeScript.3.0	TypeScript 3.0 SDK	15.0.27924.0
Microsoft.VisualStudio.Component.VC.ATL.ARM	Visual C++ ATL for ARM	15.7.27625.0

Component ID	Name	Version
Microsoft.VisualStudio.Component.VC.ATL.ARM.Spectre	Visual C++ ATL for ARM with Spectre Mitigations	15.7.27625.0
Microsoft.VisualStudio.Component.VC.ATL.ARM64	Visual C++ ATL for ARM64	15.7.27625.0
Microsoft.VisualStudio.Component.VC.ATL.ARM64.Spectre	Visual C++ ATL for ARM64 with Spectre Mitigations	15.7.27625.0
Microsoft.VisualStudio.Component.VC.ATL.Spectre	Visual C++ ATL (x86/x64) with Spectre Mitigations	15.7.27625.0
Microsoft.VisualStudio.Component.VC.ATLMFC.Spectre	Visual C++ MFC for x86/x64 with Spectre Mitigations	15.7.27625.0
Microsoft.VisualStudio.Component.VC.ClangC2	Clang/C2 (experimental)	15.7.27520.0
Microsoft.VisualStudio.Component.VC.MFC.ARM	Visual C++ MFC for ARM	15.7.27625.0
Microsoft.VisualStudio.Component.VC.MFC.ARM.Spectre	Visual C++ MFC for ARM with Spectre Mitigations	15.7.27625.0
Microsoft.VisualStudio.Component.VC.MFC.ARM64	Visual C++ MFC for ARM64	15.7.27625.0
Microsoft.VisualStudio.Component.VC.MFC.ARM64.Spectre	Visual C++ MFC support for ARM64 with Spectre Mitigations	15.7.27625.0
Microsoft.VisualStudio.Component.VC.Runtimes.ARM.Spectre	VC++ 2017 version 15.9 v14.16 Libs for Spectre (ARM)	15.9.28230.55
Microsoft.VisualStudio.Component.VC.Runtimes.ARM64.Spectre	VC++ 2017 version 15.9 v14.16 Libs for Spectre (ARM64)	15.9.28230.55
Microsoft.VisualStudio.Component.VC.Runtimes.x86.x64.Spectre	VC++ 2017 version 15.9 v14.16 Libs for Spectre (x86 and x64)	15.9.28230.55
Microsoft.VisualStudio.Component.VC.Tools.14.11	VC++ 2017 version 15.4 v14.11 toolset	15.0.27924.0
Microsoft.VisualStudio.Component.VC.Tools.14.12	VC++ 2017 version 15.5 v14.12 toolset	15.0.27924.0
Microsoft.VisualStudio.Component.VC.Tools.14.13	VC++ 2017 version 15.6 v14.13 toolset	15.0.27924.0
Microsoft.VisualStudio.Component.VC.Tools.14.14	VC++ 2017 version 15.7 v14.14 toolset	15.0.27924.0
Microsoft.VisualStudio.Component.VC.Tools.14.15	VC++ 2017 version 15.8 v14.15 toolset	15.0.28230.55

When you set dependencies in your VSIX manifest, you must specify Component IDs only. Use the tables on this page to determine the minimum component dependencies. In some scenarios, this might mean that you specify only one component from a workload. In other scenarios, it might mean that you specify multiple components from a single workload or multiple components from multiple workloads. For more information, see the [How to: Migrate extensibility projects to Visual Studio](#) page.

For more information about how to use these IDs, see the [Use command-line parameters to install Visual Studio](#) page. And, for a list of workload and component IDs for other products, see the [Visual Studio workload and component IDs](#) page.

To install these components by using the manual installer, download and run the [Build Tools for Visual Studio](#).

Azure development build tools

ID: Microsoft.VisualStudio.Workload.AzureBuildTools

Description: MSBuild tasks and targets for building Azure applications.

Components included by this workload

COMPONENT ID	NAME	VERSION	DEPENDENCY TYPE
Microsoft.Net.Component.4.7.2.TargetingPack	.NET Framework 4.7.2 targeting pack	16.0.28517.75	Required
Microsoft.Net.Component.4.8.SDK	.NET Framework 4.8 SDK	16.4.29313.120	Required
Microsoft.Net.ComponentGroup.DevelopmentPrerequisites	.NET Framework 4.7.2 development tools	16.3.29207.166	Required
Microsoft.VisualStudio.Component.Azure.AuthoringTools	Azure Authoring Tools	16.0.28625.61	Required
Microsoft.VisualStudio.Component.Azure.ClientLibs	Azure libraries for .NET	16.0.28315.86	Required
Microsoft.VisualStudio.Component.Azure.Waverton.BuildTools	Azure Cloud Services build tools	16.3.29207.166	Required
Microsoft.VisualStudio.Component.DockerTools.BuildTools	Container development tools - Build Tools	16.0.28625.61	Required
Microsoft.VisualStudio.Component.NuGet.BuildTools	NuGet targets and build tasks	16.1.28829.92	Required
Microsoft.VisualStudio.Component.TypeScript.3.7	TypeScript 3.7 SDK	16.0.29429.68	Required
Microsoft.VisualStudio.Wcf.BuildTools.ComponentGroup	Windows Communication Foundation build tools	16.0.28516.191	Required

Component ID	Name	Version	Dependency Type
Microsoft.VisualStudio.Web.BuildTools.ComponentGroup	Web development build tools	16.0.28516.191	Required
Microsoft.Net.Component.4.5.1.TargetingPack	.NET Framework 4.5.1 targeting pack	16.0.28517.75	Recommended
Microsoft.Net.Component.4.5.2.TargetingPack	.NET Framework 4.5.2 targeting pack	16.0.28517.75	Recommended
Microsoft.Net.Component.4.5.TargetingPack	.NET Framework 4.5 targeting pack	16.0.28517.75	Recommended
Microsoft.Net.Component.4.6.TargetingPack	.NET Framework 4.6 targeting pack	16.0.28517.75	Recommended
Microsoft.Net.Component.4.TargetingPack	.NET Framework 4 targeting pack	16.0.28517.75	Recommended
Microsoft.Net.ComponentGroup.TargetingPacks.Common	.NET Framework 4 – 4.6 development tools	16.0.28516.191	Recommended
Microsoft.VisualStudio.Component.AspNet45	Advanced ASP.NET features	16.0.28315.86	Recommended
Microsoft.VisualStudio.Component.WebDeploy	Web Deploy	16.0.28517.75	Recommended
Microsoft.Net.Component.3.5.DeveloperTools	.NET Framework 3.5 development tools	16.0.28517.75	Optional
Microsoft.Net.Component.4.6.1.TargetingPack	.NET Framework 4.6.1 targeting pack	16.0.28517.75	Optional
Microsoft.Net.Component.4.6.2.TargetingPack	.NET Framework 4.6.2 targeting pack	16.0.28517.75	Optional
Microsoft.Net.Component.4.7.1.TargetingPack	.NET Framework 4.7.1 targeting pack	16.0.28517.75	Optional
Microsoft.Net.Component.4.7.TargetingPack	.NET Framework 4.7 targeting pack	16.0.28517.75	Optional
Microsoft.Net.Component.4.8.TargetingPack	.NET Framework 4.8 targeting pack	16.4.29313.120	Optional
Microsoft.Net.ComponentGroup.4.6.1.DeveloperTools	.NET Framework 4.6.1 development tools	16.3.29207.166	Optional
Microsoft.Net.ComponentGroup.4.6.2.DeveloperTools	.NET Framework 4.6.2 development tools	16.3.29207.166	Optional

Component ID	Name	Version	Dependency Type
Microsoft.Net.ComponentGroup.4.7.1.DeveloperTools	.NET Framework 4.7.1 development tools	16.3.29207.166	Optional
Microsoft.Net.ComponentGroup.4.7.DeveloperTools	.NET Framework 4.7 development tools	16.3.29207.166	Optional
Microsoft.Net.ComponentGroup.4.8.DeveloperTools	.NET Framework 4.8 development tools	16.4.29318.151	Optional
Microsoft.Net.Core.Component.SDK.2.1	.NET Core 2.1 LTS Runtime	16.4.29519.181	Optional
Microsoft.NetCore.Component.SDK	.NET Core 3.1 SDK	16.4.29519.181	Optional

Data storage and processing build tools

ID: Microsoft.VisualStudio.Workload.DataBuildTools

Description: Build SQL Server Database Projects

Components included by this workload

Component ID	Name	Version	Dependency Type
Microsoft.Net.Component.4.5.1.TargetingPack	.NET Framework 4.5.1 targeting pack	16.0.28517.75	Recommended
Microsoft.Net.Component.4.5.2.TargetingPack	.NET Framework 4.5.2 targeting pack	16.0.28517.75	Recommended
Microsoft.Net.Component.4.5.TargetingPack	.NET Framework 4.5 targeting pack	16.0.28517.75	Recommended
Microsoft.Net.Component.4.6.TargetingPack	.NET Framework 4.6 targeting pack	16.0.28517.75	Recommended
Microsoft.Net.Component.4.TargetingPack	.NET Framework 4 targeting pack	16.0.28517.75	Recommended
Microsoft.Net.ComponentGroup.TargetingPacks.Common	.NET Framework 4 – 4.6 development tools	16.0.28516.191	Recommended
Microsoft.VisualStudio.Component.Roslyn.Compiler	C# and Visual Basic Roslyn compilers	16.0.28714.129	Recommended
Microsoft.VisualStudio.Component.Roslyn.LanguageServices	C# and Visual Basic	16.4.29429.68	Recommended
Microsoft.VisualStudio.Component.SQL.SSDTBuildSku	SQL Server Data Tools - Build Tools	16.0.28315.86	Recommended

.NET desktop build tools

ID: Microsoft.VisualStudio.Workload.ManagedDesktopBuildTools

Description: Tools for building WPF, Windows Forms, and console applications using C#, Visual Basic, and F#.

Components included by this workload

Component ID	Name	Version	Dependency Type
Microsoft.Component.MSBuild	MSBuild	16.4.29429.68	Required
Microsoft.Net.Component.4.6.1.TargetingPack	.NET Framework 4.6.1 targeting pack	16.0.28517.75	Required
Microsoft.Net.Component.4.8.SDK	.NET Framework 4.8 SDK	16.4.29313.120	Required
Microsoft.VisualStudio.Component.NuGet.BuildTools	NuGet targets and build tasks	16.1.28829.92	Required
Microsoft.VisualStudio.Component.Roslyn.Compiler	C# and Visual Basic Roslyn compilers	16.0.28714.129	Required
Microsoft.Component.ClickOnce.MSBuild	ClickOnce Build Tools	16.0.28625.61	Recommended
Microsoft.Net.Component.4.5.1.TargetingPack	.NET Framework 4.5.1 targeting pack	16.0.28517.75	Recommended
Microsoft.Net.Component.4.5.2.TargetingPack	.NET Framework 4.5.2 targeting pack	16.0.28517.75	Recommended
Microsoft.Net.Component.4.5.TargetingPack	.NET Framework 4.5 targeting pack	16.0.28517.75	Recommended
Microsoft.Net.Component.4.6.TargetingPack	.NET Framework 4.6 targeting pack	16.0.28517.75	Recommended
Microsoft.Net.Component.4.TargetingPack	.NET Framework 4 targeting pack	16.0.28517.75	Recommended
Microsoft.Net.ComponentGroup.TargetingPacks.Common	.NET Framework 4 – 4.6 development tools	16.0.28516.191	Recommended
Microsoft.Net.Core.Component.SDK.2.1	.NET Core 2.1 LTS Runtime	16.4.29519.181	Recommended
Microsoft.NetCore.Component.SDK	.NET Core 3.1 SDK	16.4.29519.181	Recommended
Microsoft.VisualStudio.Component.TestTools.BuildTools	Testing tools core features - Build Tools	16.4.29409.204	Recommended

Component ID	Name	Version	Dependency Type
Microsoft.VisualStudio.Wcf.BuildTools.ComponentGroup	Windows Communication Foundation build tools	16.0.28516.191	Recommended
Microsoft.Net.Component.3.5.DeveloperTools	.NET Framework 3.5 development tools	16.0.28517.75	Optional
Microsoft.Net.Component.4.6.2.TargetingPack	.NET Framework 4.6.2 targeting pack	16.0.28517.75	Optional
Microsoft.Net.Component.4.7.1.TargetingPack	.NET Framework 4.7.1 targeting pack	16.0.28517.75	Optional
Microsoft.Net.Component.4.7.TargetingPack	.NET Framework 4.7 targeting pack	16.0.28517.75	Optional
Microsoft.Net.Component.4.8.TargetingPack	.NET Framework 4.8 targeting pack	16.4.29313.120	Optional
Microsoft.Net.ComponentGroup.4.6.1.DeveloperTools	.NET Framework 4.6.1 development tools	16.3.29207.166	Optional
Microsoft.Net.ComponentGroup.4.6.2.DeveloperTools	.NET Framework 4.6.2 development tools	16.3.29207.166	Optional
Microsoft.Net.ComponentGroup.4.7.1.DeveloperTools	.NET Framework 4.7.1 development tools	16.3.29207.166	Optional
Microsoft.Net.ComponentGroup.4.7.DeveloperTools	.NET Framework 4.7 development tools	16.3.29207.166	Optional
Microsoft.Net.ComponentGroup.4.8.DeveloperTools	.NET Framework 4.8 development tools	16.4.29318.151	Optional
Microsoft.VisualStudio.Component.FSharp.MSBuild	F# compiler	16.0.28528.71	Optional

MSBuild Tools

ID: Microsoft.VisualStudio.Workload.MSBuildTools

Description: Provides the tools required to build MSBuild-based applications.

Components included by this workload

Component ID	Name	Version	Dependency Type
Microsoft.Component.MSBuild	MSBuild	16.4.29429.68	Required
Microsoft.VisualStudio.Component.CoreBuildTools	Visual Studio Build Tools Core	16.0.28315.86	Required
Microsoft.VisualStudio.Component.Roslyn.Compiler	C# and Visual Basic Roslyn compilers	16.0.28714.129	Required

.NET Core build tools

ID: Microsoft.VisualStudio.Workload.NetCoreBuildTools

Description: Tools for building applications using .NET Core, ASP.NET Core, HTML/JavaScript, and Containers.

Components included by this workload

COMPONENT ID	NAME	VERSION	DEPENDENCY TYPE
Microsoft.NetCore.BuildTools.ComponentGroup	.NET Core build tools	16.3.29102.218	Required
Microsoft.NetCore.Component.SDK	.NET Core 3.1 SDK	16.4.29519.181	Required
Microsoft.VisualStudio.Component.NuGet.BuildTools	NuGet targets and build tasks	16.1.28829.92	Required
Microsoft.Net.Core.Component.SDK.2.1	.NET Core 2.1 LTS Runtime	16.4.29519.181	Recommended

Node.js build tools

ID: Microsoft.VisualStudio.Workload.NodeBuildTools

Description: MSBuild tasks and targets for building scalable network applications using Node.js, an asynchronous event-driven JavaScript runtime.

Components included by this workload

COMPONENT ID	NAME	VERSION	DEPENDENCY TYPE
Microsoft.VisualStudio.Component.Node.Build	Node.js MSBuild support	16.0.28517.75	Required
Microsoft.VisualStudio.Component.TypeScript.3.7	TypeScript 3.7 SDK	16.0.29429.68	Required

Office/SharePoint build tools

ID: Microsoft.VisualStudio.Workload.OfficeBuildTools

Description: Build Office and SharePoint add-ins, and VSTO add-ins.

Components included by this workload

COMPONENT ID	NAME	VERSION	DEPENDENCY TYPE
Microsoft.Component.ClickOnce.MSBuild	ClickOnce Build Tools	16.0.28625.61	Required
Microsoft.Component.MSBuild	MSBuild	16.4.29429.68	Required
Microsoft.Net.Component.4.5.2.TargetingPack	.NET Framework 4.5.2 targeting pack	16.0.28517.75	Required

Component ID	Name	Version	Dependency Type
Microsoft.Net.Component.4.5.TargetingPack	.NET Framework 4.5 targeting pack	16.0.28517.75	Required
Microsoft.Net.Component.4.6.1.TargetingPack	.NET Framework 4.6.1 targeting pack	16.0.28517.75	Required
Microsoft.Net.Component.4.7.2.TargetingPack	.NET Framework 4.7.2 targeting pack	16.0.28517.75	Required
Microsoft.Net.Component.4.8.SDK	.NET Framework 4.8 SDK	16.4.29313.120	Required
Microsoft.Net.Component.4.TargetingPack	.NET Framework 4 targeting pack	16.0.28517.75	Required
Microsoft.Net.ComponentGroup.DevelopmentPrerequisites	.NET Framework 4.7.2 development tools	16.3.29207.166	Required
Microsoft.VisualStudio.Component.NuGet	NuGet package manager	16.1.28829.92	Required
Microsoft.VisualStudio.Component.NuGet.BuildTools	NuGet targets and build tasks	16.1.28829.92	Required
Microsoft.VisualStudio.Component.Roslyn.Compiler	C# and Visual Basic Roslyn compilers	16.0.28714.129	Required
Microsoft.VisualStudio.Component.Sharepoint.BuildTools	Office/SharePoint development build tools	16.0.28625.61	Required
Microsoft.VisualStudio.Component.Workflow.BuildTools	Windows Workflow Foundation Build Tools	16.0.28315.86	Required
Microsoft.VisualStudio.Wcf.BuildTools.ComponentGroup	Windows Communication Foundation build tools	16.0.28516.191	Required
Microsoft.VisualStudio.Web.BuildTools.ComponentGroup	Web development build tools	16.0.28516.191	Required
Microsoft.VisualStudio.Component.TeamOffice.BuildTools	Visual Studio Tools for Office (VSTO) build tools	16.0.28625.61	Recommended
Microsoft.VisualStudio.Component.WebDeploy	Web Deploy	16.0.28517.75	Recommended
Microsoft.Net.Component.4.6.2.TargetingPack	.NET Framework 4.6.2 targeting pack	16.0.28517.75	Optional
Microsoft.Net.Component.4.7.1.TargetingPack	.NET Framework 4.7.1 targeting pack	16.0.28517.75	Optional

Component ID	Name	Version	Dependency Type
Microsoft.Net.Component.4.7.TargetingPack	.NET Framework 4.7 targeting pack	16.0.28517.75	Optional
Microsoft.Net.Component.4.8.TargetingPack	.NET Framework 4.8 targeting pack	16.4.29313.120	Optional
Microsoft.Net.ComponentGroup.4.6.1.DeveloperTools	.NET Framework 4.6.1 development tools	16.3.29207.166	Optional
Microsoft.Net.ComponentGroup.4.6.2.DeveloperTools	.NET Framework 4.6.2 development tools	16.3.29207.166	Optional
Microsoft.Net.ComponentGroup.4.7.1.DeveloperTools	.NET Framework 4.7.1 development tools	16.3.29207.166	Optional
Microsoft.Net.ComponentGroup.4.7.2.DeveloperTools	.NET Framework 4.7 development tools	16.3.29207.166	Optional
Microsoft.Net.ComponentGroup.4.8.DeveloperTools	.NET Framework 4.8 development tools	16.4.29318.151	Optional

Universal Windows Platform build tools

ID: Microsoft.VisualStudio.Workload.UniversalBuildTools

Description: Provides the tools required to build Universal Windows Platform applications.

Components included by this workload

Component ID	Name	Version	Dependency Type
Microsoft.Component.MSBuild	MSBuild	16.4.29429.68	Required
Microsoft.Component.NetFX.Native	.NET Native	16.4.29429.68	Required
Microsoft.Net.Component.4.8.SDK	.NET Framework 4.8 SDK	16.4.29313.120	Required
Microsoft.NetCore.Component.SDK	.NET Core 3.1 SDK	16.4.29519.181	Required
Microsoft.VisualStudio.Component.NuGet.BuildTools	NuGet targets and build tasks	16.1.28829.92	Required
Microsoft.VisualStudio.Component.Roslyn.Compiler	C# and Visual Basic Roslyn compilers	16.0.28714.129	Required
Microsoft.VisualStudio.ComponentGroup.UWP.BuildTools	Universal Windows Platform build prerequisites	16.3.29207.166	Required

Component ID	Name	Version	Dependency Type
Microsoft.VisualStudio.Component.Windows10SDK.18362	Windows 10 SDK (10.0.18362.0)	16.1.28829.92	Recommended
Microsoft.Net.Component.4.7.2.SDK	.NET Framework 4.7.2 SDK	16.4.29409.204	Optional
Microsoft.VisualStudio.Component.TextTemplating	Text Template Transformation	16.0.28625.61	Optional
Microsoft.VisualStudio.Component.UWP.VC.ARM64	C++ Universal Windows Platform support for v142 build tools (ARM64)	16.3.29207.166	Optional
Microsoft.VisualStudio.Component.VC.CoreLde	C++ core features	16.0.28625.61	Optional
Microsoft.VisualStudio.Component.VC.Redist.14.Latest	C++ 2019 Redistributable Update	16.4.29429.68	Optional
Microsoft.VisualStudio.Component.VC.Tools.ARM	MSVC v142 - VS 2019 C++ ARM build tools (v14.24)	16.4.29409.204	Optional
Microsoft.VisualStudio.Component.VC.Tools.ARM64	MSVC v142 - VS 2019 C++ ARM64 build tools (v14.24)	16.4.29409.204	Optional
Microsoft.VisualStudio.Component.VC.Tools.x86.x64	MSVC v142 - VS 2019 C++ x64/x86 build tools (v14.24)	16.4.29409.204	Optional
Microsoft.VisualStudio.Component.VC.v141.ARM	MSVC v141 - VS 2017 C++ ARM build tools (v14.16)	16.2.29003.222	Optional
Microsoft.VisualStudio.Component.VC.v141.ARM64	MSVC v141 - VS 2017 C++ ARM64 build tools (v14.16)	16.1.28829.92	Optional
Microsoft.VisualStudio.Component.VC.v141.x86.x64	MSVC v141 - VS 2017 C++ x64/x86 build tools (v14.16)	16.1.28829.92	Optional
Microsoft.VisualStudio.Component.Windows10SDK.16299	Windows 10 SDK (10.0.16299.0)	16.0.28517.75	Optional
Microsoft.VisualStudio.Component.Windows10SDK.17134	Windows 10 SDK (10.0.17134.0)	16.0.28517.75	Optional
Microsoft.VisualStudio.Component.Windows10SDK.17763	Windows 10 SDK (10.0.17763.0)	16.0.28517.75	Optional
Microsoft.VisualStudio.ComponentGroup.NativeDesktop.Core	C++ core desktop features	16.2.29012.281	Optional

Component ID	Name	Version	Dependency Type
Microsoft.VisualStudio.ComponentGroup.UWP.VC.BuildTools	C++ (v142) Universal Windows Platform tools	16.3.29207.166	Optional
Microsoft.VisualStudio.ComponentGroup.UWP.VC.v141.BuildTools	C++ (v141) Universal Windows Platform tools	16.3.29207.166	Optional

C++ build tools

ID: Microsoft.VisualStudio.Workload.VCTools

Description: Build Windows desktop applications using the Microsoft C++ toolset, ATL, or MFC.

Components included by this workload

Component ID	Name	Version	Dependency Type
Microsoft.VisualStudio.Component.VC.CoreBuildTools	C++ Build Tools core features	16.0.28625.61	Required
Microsoft.VisualStudio.Component.VC.Redist.14.Latest	C++ 2019 Redistributable Update	16.4.29429.68	Required
Microsoft.VisualStudio.Component.Windows10SDK	Windows Universal C Runtime	16.4.29409.204	Required
Microsoft.VisualStudio.Component.TestTools.BuildTools	Testing tools core features - Build Tools	16.4.29409.204	Recommended
Microsoft.VisualStudio.Component.VC.ASAN	C++ AddressSanitizer (Experimental)	16.4.29429.68	Recommended
Microsoft.VisualStudio.Component.VC.CMake.Project	C++ CMake tools for Windows	16.3.29103.31	Recommended
Microsoft.VisualStudio.Component.VC.Tools.x86.x64	MSVC v142 - VS 2019 C++ x64/x86 build tools (v14.24)	16.4.29409.204	Recommended
Microsoft.VisualStudio.Component.Windows10SDK.18362	Windows 10 SDK (10.0.18362.0)	16.1.28829.92	Recommended
Microsoft.Component.MSBuild	MSBuild	16.4.29429.68	Optional
Microsoft.Component.VC.Runtime.UCRTSDK	Windows Universal CRT SDK	16.0.28625.61	Optional
Microsoft.Net.Component.4.6.1.TargetingPack	.NET Framework 4.6.1 targeting pack	16.0.28517.75	Optional
Microsoft.Net.Component.4.8.SDK	.NET Framework 4.8 SDK	16.4.29313.120	Optional

Component ID	Name	Version	Dependency Type
Microsoft.VisualStudio.Component.Roslyn.Compiler	C# and Visual Basic Roslyn compilers	16.0.28714.129	Optional
Microsoft.VisualStudio.Component.TextTemplating	Text Template Transformation	16.0.28625.61	Optional
Microsoft.VisualStudio.Component.VC.140	MSVC v140 - VS 2015 C++ build tools (v14.00)	16.0.28625.61	Optional
Microsoft.VisualStudio.Component.VC.ATL	C++ ATL for latest v142 build tools (x86 & x64)	16.4.29313.120	Optional
Microsoft.VisualStudio.Component.VC.ATLMFC	C++ MFC for latest v142 build tools (x86 & x64)	16.4.29313.120	Optional
Microsoft.VisualStudio.Component.VC.CLI.Support	C++/CLI support for v142 build tools (14.24)	16.4.29409.204	Optional
Microsoft.VisualStudio.Component.VC.CoreCRT	C++ core features	16.0.28625.61	Optional
Microsoft.VisualStudio.Component.VC.Llvm.Clang	C++ Clang Compiler for Windows (9.0.0)	16.4.29511.114	Optional
Microsoft.VisualStudio.Component.VC.Llvm.ClangToolset	C++ Clang-cl for v142 build tools (x64/x86)	16.3.29207.166	Optional
Microsoft.VisualStudio.Component.VC.Modules.x86.x64	C++ Modules for v142 build tools (x64/x86 – experimental)	16.0.28625.61	Optional
Microsoft.VisualStudio.Component.VC.v141.x86.x64	MSVC v141 - VS 2017 C++ x64/x86 build tools (v14.16)	16.1.28829.92	Optional
Microsoft.VisualStudio.Component.Windows10SDK.16299	Windows 10 SDK (10.0.16299.0)	16.0.28517.75	Optional
Microsoft.VisualStudio.Component.Windows10SDK.17134	Windows 10 SDK (10.0.17134.0)	16.0.28517.75	Optional
Microsoft.VisualStudio.Component.Windows10SDK.17763	Windows 10 SDK (10.0.17763.0)	16.0.28517.75	Optional
Microsoft.VisualStudio.ComponentGroup.NativeDesktop.Core	C++ core desktop features	16.2.29012.281	Optional
Microsoft.VisualStudio.ComponentGroup.NativeDesktop.Llvm.Clang	C++ Clang tools for Windows (9.0.0 - x64/x86)	16.4.29511.114	Optional

Visual Studio extension development

ID: Microsoft.VisualStudio.Workload.VisualStudioExtensionBuildTools

Description: Tools for building add-ons and extensions for Visual Studio, including new commands, code analyzers and tool windows.

Components included by this workload

Component ID	Name	Version	Dependency Type
Microsoft.Component.MSBuild	MSBuild	16.4.29429.68	Required
Microsoft.Net.Component.4.6.TargetingPack	.NET Framework 4.6 targeting pack	16.0.28517.75	Required
Microsoft.Net.Component.4.7.2.TargetingPack	.NET Framework 4.7.2 targeting pack	16.0.28517.75	Required
Microsoft.Net.Component.4.8.SDK	.NET Framework 4.8 SDK	16.4.29313.120	Required
Microsoft.Net.ComponentGroup.DevelopmentPrerequisites	.NET Framework 4.7.2 development tools	16.3.29207.166	Required
Microsoft.VisualStudio.Component.NuGet.BuildTools	NuGet targets and build tasks	16.1.28829.92	Required
Microsoft.VisualStudio.Component.Roslyn.Compiler	C# and Visual Basic Roslyn compilers	16.0.28714.129	Required
Microsoft.VisualStudio.Component.VSSDKBuildTools	Visual Studio SDK Build Tools Core	16.0.28315.86	Required
Microsoft.VisualStudio.ComponentGroup.VisualStudioExtensionBuildTools.Prerequisites	Visual Studio extension development prerequisites	16.4.29318.151	Required
Component.Dotfuscator	PreEmptive Protection - Dotfuscator	16.0.28528.71	Optional
Microsoft.Component.VC_Runtime.OSSupport	C++ Universal Windows Platform runtime for v142 build tools	16.4.29409.204	Optional
Microsoft.VisualStudio.Component.VC.ATL	C++ ATL for latest v142 build tools (x86 & x64)	16.4.29313.120	Optional
Microsoft.VisualStudio.Component.VC.ATLMFC	C++ MFC for latest v142 build tools (x86 & x64)	16.4.29313.120	Optional
Microsoft.VisualStudio.Component.VC.Tools.x86.x64	MSVC v142 - VS 2019 C++ x64/x86 build tools (v14.24)	16.4.29409.204	Optional

Web development build tools

ID: Microsoft.VisualStudio.Workload.WebBuildTools

Description: MSBuild tasks and targets for building web applications.

Components included by this workload

Component ID	Name	Version	Dependency Type
Microsoft.Net.Component.4.7.2.TargetingPack	.NET Framework 4.7.2 targeting pack	16.0.28517.75	Required
Microsoft.Net.Component.4.8.SDK	.NET Framework 4.8 SDK	16.4.29313.120	Required
Microsoft.Net.ComponentGroup.DevelopmentPrerequisites	.NET Framework 4.7.2 development tools	16.3.29207.166	Required
Microsoft.VisualStudio.Component.NuGet.BuildTools	NuGet targets and build tasks	16.1.28829.92	Required
Microsoft.VisualStudio.Component.TypeScript.3.7	TypeScript 3.7 SDK	16.0.29429.68	Required
Microsoft.VisualStudio.Web.BuildTools.ComponentGroup	Web development build tools	16.0.28516.191	Required
Microsoft.Component.ClickOnce.MSBuild	ClickOnce Build Tools	16.0.28625.61	Recommended
Microsoft.Net.Component.4.5.1.TargetingPack	.NET Framework 4.5.1 targeting pack	16.0.28517.75	Recommended
Microsoft.Net.Component.4.5.2.TargetingPack	.NET Framework 4.5.2 targeting pack	16.0.28517.75	Recommended
Microsoft.Net.Component.4.5.TargetingPack	.NET Framework 4.5 targeting pack	16.0.28517.75	Recommended
Microsoft.Net.Component.4.6.TargetingPack	.NET Framework 4.6 targeting pack	16.0.28517.75	Recommended
Microsoft.Net.Component.4.TargetingPack	.NET Framework 4 targeting pack	16.0.28517.75	Recommended
Microsoft.Net.ComponentGroup.TargetingPacks.Common	.NET Framework 4 – 4.6 development tools	16.0.28516.191	Recommended
Microsoft.Net.Core.Component.SDK.2.1	.NET Core 2.1 LTS Runtime	16.4.29519.181	Recommended
Microsoft.NetCore.Component.SDK	.NET Core 3.1 SDK	16.4.29519.181	Recommended

Component ID	Name	Version	Dependency Type
Microsoft.VisualStudio.Component.AspNet45	Advanced ASP.NET features	16.0.28315.86	Recommended
Microsoft.VisualStudio.Component.DockerTools.BuildTools	Container development tools - Build Tools	16.0.28625.61	Recommended
Microsoft.VisualStudio.Component.TestTools.BuildTools	Testing tools core features - Build Tools	16.4.29409.204	Recommended
Microsoft.VisualStudio.Component.WebDeploy	Web Deploy	16.0.28517.75	Recommended
Microsoft.VisualStudio.Wcf.BuildTools.ComponentGroup	Windows Communication Foundation build tools	16.0.28516.191	Recommended
Microsoft.Net.Component.3.5DeveloperTools	.NET Framework 3.5 development tools	16.0.28517.75	Optional
Microsoft.Net.Component.4.6.1.TargetingPack	.NET Framework 4.6.1 targeting pack	16.0.28517.75	Optional
Microsoft.Net.Component.4.6.2.TargetingPack	.NET Framework 4.6.2 targeting pack	16.0.28517.75	Optional
Microsoft.Net.Component.4.7.1.TargetingPack	.NET Framework 4.7.1 targeting pack	16.0.28517.75	Optional
Microsoft.Net.Component.4.7.TargetingPack	.NET Framework 4.7 targeting pack	16.0.28517.75	Optional
Microsoft.Net.Component.4.8.TargetingPack	.NET Framework 4.8 targeting pack	16.4.29313.120	Optional
Microsoft.Net.ComponentGroup.4.6.1.DeveloperTools	.NET Framework 4.6.1 development tools	16.3.29207.166	Optional
Microsoft.Net.ComponentGroup.4.6.2.DeveloperTools	.NET Framework 4.6.2 development tools	16.3.29207.166	Optional
Microsoft.Net.ComponentGroup.4.7.1.DeveloperTools	.NET Framework 4.7.1 development tools	16.3.29207.166	Optional
Microsoft.Net.ComponentGroup.4.7.DeveloperTools	.NET Framework 4.7 development tools	16.3.29207.166	Optional
Microsoft.Net.ComponentGroup.4.8.DeveloperTools	.NET Framework 4.8 development tools	16.4.29318.151	Optional

Mobile Development with .NET

ID: Microsoft.VisualStudio.Workload.XamarinBuildTools

Description: Tools for building cross-platform applications for iOS, Android and Windows using C# and F#.

Components included by this workload

Component ID	Name	Version	Dependency Type
Microsoft.Component.MSBuild	MSBuild	16.4.29429.68	Required
Microsoft.Net.Component.4.6.1.TargetingPack	.NET Framework 4.6.1 targeting pack	16.0.28517.75	Required
Microsoft.Net.Component.4.8.SDK	.NET Framework 4.8 SDK	16.4.29313.120	Required
Microsoft.VisualStudio.Component.NuGet.BuildTools	NuGet targets and build tasks	16.1.28829.92	Required
Microsoft.VisualStudio.Component.Roslyn.Compiler	C# and Visual Basic Roslyn compilers	16.0.28714.129	Required
Component.Android.SDK28	Android SDK setup (API level 28)	16.2.29003.222	Optional
Component.OpenJDK	OpenJDK (Microsoft distribution)	16.1.28811.260	Optional

Unaffiliated components

These are components that are not included with any workload, but may be selected as an individual component.

Component ID	Name	Version
Microsoft.Net.Core.Component.SDK.2.2	.NET Core 2.2 Runtime	16.4.29519.181
Microsoft.Net.Core.Component.SDK.3.0	.NET Core 3.0 Runtime	16.4.29519.181
Microsoft.VisualStudio.Component.VC.14.20.ARM	MSVC v142 - VS 2019 C++ ARM build tools (v14.20)	16.4.29511.114
Microsoft.VisualStudio.Component.VC.14.20.ARM.Spectre	MSVC v142 - VS 2019 C++ ARM Spectre-mitigated libs (v14.20)	16.4.29511.114
Microsoft.VisualStudio.Component.VC.14.20.ARM64	MSVC v142 - VS 2019 C++ ARM64 build tools (v14.20)	16.4.29511.114
Microsoft.VisualStudio.Component.VC.14.20.ARM64.Spectre	MSVC v142 - VS 2019 C++ ARM64 Spectre-mitigated libs (v14.20)	16.4.29511.114
Microsoft.VisualStudio.Component.VC.14.20.ATL	C++ v14.20 ATL for v142 build tools (x86 & x64)	16.1.28829.92
Microsoft.VisualStudio.Component.VC.14.20.ATL.ARM	C++ v14.20 ATL for v142 build tools (ARM)	16.4.29511.114
Microsoft.VisualStudio.Component.VC.14.20.ATL.ARM.Spectre	C++ v14.20 ATL for v142 build tools with Spectre Mitigations (ARM)	16.4.29511.114

Component ID	Name	Version
Microsoft.VisualStudio.Component.VC.14.20.ATL.ARM64	C++ v14.20 ATL for v142 build tools (ARM64)	16.4.29511.114
Microsoft.VisualStudio.Component.VC.14.20.ATL.ARM64.Spectre	C++ v14.20 ATL for v142 build tools with Spectre Mitigations (ARM64)	16.4.29511.114
Microsoft.VisualStudio.Component.VC.14.20.ATL.Spectre	C++ v14.20 ATL for v142 build tools with Spectre Mitigations (x86 & x64)	16.4.29511.114
Microsoft.VisualStudio.Component.VC.14.20.CLI.Support	C++/CLI support for v142 build tools (14.20)	16.4.29409.204
Microsoft.VisualStudio.Component.VC.14.20.MFC	C++ v14.20 MFC for v142 build tools (x86 & x64)	16.2.29003.222
Microsoft.VisualStudio.Component.VC.14.20.MFC.ARM	C++ v14.20 MFC for v142 build tools (ARM)	16.4.29511.114
Microsoft.VisualStudio.Component.VC.14.20.MFC.ARM.Spectre	C++ v14.20 MFC for v142 build tools with Spectre Mitigations (ARM)	16.4.29511.114
Microsoft.VisualStudio.Component.VC.14.20.MFC.ARM64	C++ v14.20 MFC for v142 build tools (ARM64)	16.4.29511.114
Microsoft.VisualStudio.Component.VC.14.20.MFC.ARM64.Spectre	C++ v14.20 MFC for v142 build tools with Spectre Mitigations (ARM64)	16.4.29511.114
Microsoft.VisualStudio.Component.VC.14.20.MFC.Spectre	C++ v14.20 MFC for v142 build tools with Spectre Mitigations (x86 & x64)	16.4.29511.114
Microsoft.VisualStudio.Component.VC.14.20.x86.x64	MSVC v142 - VS 2019 C++ x64/x86 build tools (v14.20)	16.4.29511.114
Microsoft.VisualStudio.Component.VC.14.20.x86.x64.Spectre	MSVC v142 - VS 2019 C++ x64/x86 Spectre-mitigated libs (v14.20)	16.4.29511.114
Microsoft.VisualStudio.Component.VC.14.21.ARM	MSVC v142 - VS 2019 C++ ARM build tools (v14.21)	16.3.29207.166
Microsoft.VisualStudio.Component.VC.14.21.ARM.Spectre	MSVC v142 - VS 2019 C++ ARM Spectre-mitigated libs (v14.21)	16.4.29429.68
Microsoft.VisualStudio.Component.VC.14.21.ARM64	MSVC v142 - VS 2019 C++ ARM64 build tools (v14.21)	16.3.29207.166
Microsoft.VisualStudio.Component.VC.14.21.ARM64.Spectre	MSVC v142 - VS 2019 C++ ARM64 Spectre-mitigated libs (v14.21)	16.4.29429.68
Microsoft.VisualStudio.Component.VC.14.21.ATL	C++ v14.21 ATL for v142 build tools (x86 & x64)	16.2.29019.55
Microsoft.VisualStudio.Component.VC.14.21.ATL.ARM	C++ v14.21 ATL for v142 build tools (ARM)	16.2.29019.55

Component ID	Name	Version
Microsoft.VisualStudio.Component.VC.14.21.ATL.ARM.Spectre	C++ v14.21 ATL for v142 build tools with Spectre Mitigations (ARM)	16.4.29429.68
Microsoft.VisualStudio.Component.VC.14.21.ATL.ARM64	C++ v14.21 ATL for v142 build tools (ARM64)	16.2.29019.55
Microsoft.VisualStudio.Component.VC.14.21.ATL.ARM64.Spectre	C++ v14.21 ATL for v142 build tools with Spectre Mitigations (ARM64)	16.4.29429.68
Microsoft.VisualStudio.Component.VC.14.21.ATL.Spectre	C++ v14.21 ATL for v142 build tools with Spectre Mitigations (x86 & x64)	16.4.29429.68
Microsoft.VisualStudio.Component.VC.14.21.CLI.Support	C++/CLI support for v142 build tools (14.21)	16.3.29207.166
Microsoft.VisualStudio.Component.VC.14.21.MFC	C++ v14.21 MFC for v142 build tools (x86 & x64)	16.2.29019.55
Microsoft.VisualStudio.Component.VC.14.21.MFC.ARM	C++ v14.21 MFC for v142 build tools (ARM)	16.2.29019.55
Microsoft.VisualStudio.Component.VC.14.21.MFC.ARM.Spectre	C++ v14.21 MFC for v142 build tools with Spectre Mitigations (ARM)	16.4.29429.68
Microsoft.VisualStudio.Component.VC.14.21.MFC.ARM64	C++ v14.21 MFC for v142 build tools (ARM64)	16.2.29019.55
Microsoft.VisualStudio.Component.VC.14.21.MFC.ARM64.Spectre	C++ v14.21 MFC for v142 build tools with Spectre Mitigations (ARM64)	16.4.29429.68
Microsoft.VisualStudio.Component.VC.14.21.MFC.Spectre	C++ v14.21 MFC for v142 build tools with Spectre Mitigations (x86 & x64)	16.4.29429.68
Microsoft.VisualStudio.Component.VC.14.21.x86.x64	MSVC v142 - VS 2019 C++ x64/x86 build tools (v14.21)	16.3.29207.166
Microsoft.VisualStudio.Component.VC.14.21.x86.x64.Spectre	MSVC v142 - VS 2019 C++ x64/x86 Spectre-mitigated libs (v14.21)	16.4.29429.68
Microsoft.VisualStudio.Component.VC.14.22.ARM	MSVC v142 - VS 2019 C++ ARM build tools (v14.22)	16.4.29313.120
Microsoft.VisualStudio.Component.VC.14.22.ARM.Spectre	MSVC v142 - VS 2019 C++ ARM Spectre-mitigated libs (v14.22)	16.4.29429.68
Microsoft.VisualStudio.Component.VC.14.22.ARM64	MSVC v142 - VS 2019 C++ ARM64 build tools (v14.22)	16.4.29313.120
Microsoft.VisualStudio.Component.VC.14.22.ARM64.Spectre	MSVC v142 - VS 2019 C++ ARM64 Spectre-mitigated libs (v14.22)	16.4.29429.68
Microsoft.VisualStudio.Component.VC.14.22.ATL	C++ v14.22 ATL for v142 build tools (x86 & x64)	16.4.29313.120

Component ID	Name	Version
Microsoft.VisualStudio.Component.VC.14.22.ATL.ARM	C++ v14.22 ATL for v142 build tools (ARM)	16.4.29313.120
Microsoft.VisualStudio.Component.VC.14.22.ATL.ARM.Spectre	C++ v14.22 ATL for v142 build tools with Spectre Mitigations (ARM)	16.4.29429.68
Microsoft.VisualStudio.Component.VC.14.22.ATL.ARM64	C++ v14.22 ATL for v142 build tools (ARM64)	16.4.29313.120
Microsoft.VisualStudio.Component.VC.14.22.ATL.ARM64.Spectre	C++ v14.22 ATL for v142 build tools with Spectre Mitigations (ARM64)	16.4.29429.68
Microsoft.VisualStudio.Component.VC.14.22.ATL.Spectre	C++ v14.22 ATL for v142 build tools with Spectre Mitigations (x86 & x64)	16.4.29429.68
Microsoft.VisualStudio.Component.VC.14.22.CLI.Support	C++/CLI support for v142 build tools (14.22)	16.4.29313.120
Microsoft.VisualStudio.Component.VC.14.22.MFC	C++ v14.22 MFC for v142 build tools (x86 & x64)	16.4.29313.120
Microsoft.VisualStudio.Component.VC.14.22.MFC.ARM	C++ v14.22 MFC for v142 build tools (ARM)	16.4.29313.120
Microsoft.VisualStudio.Component.VC.14.22.MFC.ARM.Spectre	C++ v14.22 MFC for v142 build tools with Spectre Mitigations (ARM)	16.4.29429.68
Microsoft.VisualStudio.Component.VC.14.22.MFC.ARM64	C++ v14.22 MFC for v142 build tools (ARM64)	16.4.29313.120
Microsoft.VisualStudio.Component.VC.14.22.MFC.ARM64.Spectre	C++ v14.22 MFC for v142 build tools with Spectre Mitigations (ARM64)	16.4.29429.68
Microsoft.VisualStudio.Component.VC.14.22.MFC.Spectre	C++ v14.22 MFC for v142 build tools with Spectre Mitigations (x86 & x64)	16.4.29429.68
Microsoft.VisualStudio.Component.VC.14.22.x86.x64	MSVC v142 - VS 2019 C++ x64/x86 build tools (v14.22)	16.4.29313.120
Microsoft.VisualStudio.Component.VC.14.22.x86.x64.Spectre	MSVC v142 - VS 2019 C++ x64/x86 Spectre-mitigated libs (v14.22)	16.4.29429.68
Microsoft.VisualStudio.Component.VC.14.23.ARM	MSVC v142 - VS 2019 C++ ARM build tools (v14.23)	16.4.29429.68
Microsoft.VisualStudio.Component.VC.14.23.ARM.Spectre	MSVC v142 - VS 2019 C++ ARM Spectre-mitigated libs (v14.23)	16.4.29429.68
Microsoft.VisualStudio.Component.VC.14.23.ARM64	MSVC v142 - VS 2019 C++ ARM64 build tools (v14.23)	16.4.29429.68
Microsoft.VisualStudio.Component.VC.14.23.ARM64.Spectre	MSVC v142 - VS 2019 C++ ARM64 Spectre-mitigated libs (v14.23)	16.4.29429.68

Component ID	Name	Version
Microsoft.VisualStudio.Component.VC.14.23.ATL	C++ v14.23 ATL for v142 build tools (x86 & x64)	16.4.29429.68
Microsoft.VisualStudio.Component.VC.14.23.ATL.ARM	C++ v14.23 ATL for v142 build tools (ARM)	16.4.29429.68
Microsoft.VisualStudio.Component.VC.14.23.ATL.ARM.Spectre	C++ v14.23 ATL for v142 build tools with Spectre Mitigations (ARM)	16.4.29429.68
Microsoft.VisualStudio.Component.VC.14.23.ATL.ARM64	C++ v14.23 ATL for v142 build tools (ARM64)	16.4.29429.68
Microsoft.VisualStudio.Component.VC.14.23.ATL.ARM64.Spectre	C++ v14.23 ATL for v142 build tools with Spectre Mitigations (ARM64)	16.4.29429.68
Microsoft.VisualStudio.Component.VC.14.23.ATL.Spectre	C++ v14.23 ATL for v142 build tools with Spectre Mitigations (x86 & x64)	16.4.29429.68
Microsoft.VisualStudio.Component.VC.14.23.CLI.Support	C++/CLI support for v142 build tools (14.23)	16.4.29409.204
Microsoft.VisualStudio.Component.VC.14.23.MFC	C++ v14.23 MFC for v142 build tools (x86 & x64)	16.4.29429.68
Microsoft.VisualStudio.Component.VC.14.23.MFC.ARM	C++ v14.23 MFC for v142 build tools (ARM)	16.4.29429.68
Microsoft.VisualStudio.Component.VC.14.23.MFC.ARM.Spectre	C++ v14.23 MFC for v142 build tools with Spectre Mitigations (ARM)	16.4.29429.68
Microsoft.VisualStudio.Component.VC.14.23.MFC.ARM64	C++ v14.23 MFC for v142 build tools (ARM64)	16.4.29429.68
Microsoft.VisualStudio.Component.VC.14.23.MFC.ARM64.Spectre	C++ v14.23 MFC for v142 build tools with Spectre Mitigations (ARM64)	16.4.29429.68
Microsoft.VisualStudio.Component.VC.14.23.MFC.Spectre	C++ v14.23 MFC for v142 build tools with Spectre Mitigations (x86 & x64)	16.4.29429.68
Microsoft.VisualStudio.Component.VC.14.23.x86.x64	MSVC v142 - VS 2019 C++ x64/x86 build tools (v14.23)	16.4.29429.68
Microsoft.VisualStudio.Component.VC.14.23.x86.x64.Spectre	MSVC v142 - VS 2019 C++ x64/x86 Spectre-mitigated libs (v14.23)	16.4.29429.68
Microsoft.VisualStudio.Component.VC.ATL.ARM	C++ ATL for latest v142 build tools (ARM)	16.4.29313.120
Microsoft.VisualStudio.Component.VC.ATL.ARM.Spectre	C++ ATL for latest v142 build tools with Spectre Mitigations (ARM)	16.4.29429.68
Microsoft.VisualStudio.Component.VC.ATL.ARM64	C++ ATL for latest v142 build tools (ARM64)	16.4.29313.120

Component ID	Name	Version
Microsoft.VisualStudio.Component.VC.ATL.ARM64.Spectre	C++ ATL for latest v142 build tools with Spectre Mitigations (ARM64)	16.4.29429.68
Microsoft.VisualStudio.Component.VC.ATL.Spectre	C++ ATL for latest v142 build tools with Spectre Mitigations (x86 & x64)	16.4.29429.68
Microsoft.VisualStudio.Component.VC.ATLMFC.Spectre	C++ MFC for latest v142 build tools with Spectre Mitigations (x86 & x64)	16.4.29429.68
Microsoft.VisualStudio.Component.VC.MFC.ARM	C++ MFC for latest v142 build tools (ARM)	16.4.29313.120
Microsoft.VisualStudio.Component.VC.MFC.ARM.Spectre	C++ MFC for latest v142 build tools with Spectre Mitigations (ARM)	16.4.29429.68
Microsoft.VisualStudio.Component.VC.MFC.ARM64	C++ MFC for latest v142 build tools (ARM64)	16.4.29313.120
Microsoft.VisualStudio.Component.VC.MFC.ARM64.Spectre	C++ MFC for latest v142 build tools with Spectre Mitigations (ARM64)	16.4.29429.68
Microsoft.VisualStudio.Component.VC.Redist.MSM	C++ 2019 Redistributable MSMs	16.4.29429.68
Microsoft.VisualStudio.Component.VC.Runtimes.ARM.Spectre	MSVC v142 - VS 2019 C++ ARM Spectre-mitigated libs (v14.24)	16.4.29429.68
Microsoft.VisualStudio.Component.VC.Runtimes.ARM64.Spectre	MSVC v142 - VS 2019 C++ ARM64 Spectre-mitigated libs (v14.24)	16.4.29429.68
Microsoft.VisualStudio.Component.VC.Runtimes.x86.x64.Spectre	MSVC v142 - VS 2019 C++ x64/x86 Spectre-mitigated libs (v14.24)	16.4.29429.68
Microsoft.VisualStudio.Component.VC.v141.ARM.Spectre	MSVC v141 - VS 2017 C++ ARM Spectre-mitigated libs (v14.16)	16.4.29429.68
Microsoft.VisualStudio.Component.VC.v141.ARM64.Spectre	MSVC v141 - VS 2017 C++ ARM64 Spectre-mitigated libs (v14.16)	16.4.29429.68
Microsoft.VisualStudio.Component.VC.v141.ATL	C++ ATL for v141 build tools (x86 & x64)	16.0.28625.61
Microsoft.VisualStudio.Component.VC.v141.ATL.ARM	C++ ATL for v141 build tools (ARM)	16.0.28625.61
Microsoft.VisualStudio.Component.VC.v141.ATL.ARM.Spectre	C++ ATL for v141 build tools with Spectre Mitigations (ARM)	16.0.28625.61
Microsoft.VisualStudio.Component.VC.v141.ATL.ARM64	C++ ATL for v141 build tools (ARM64)	16.0.28625.61
Microsoft.VisualStudio.Component.VC.v141.ATL.ARM64.Spectre	C++ ATL for v141 build tools with Spectre Mitigations (ARM64)	16.0.28625.61

Component ID	Name	Version
Microsoft.VisualStudio.Component.VC.v141.ATL.Spectre	C++ ATL for v141 build tools with Spectre Mitigations (x86 & x64)	16.0.28625.61
Microsoft.VisualStudio.Component.VC.v141.CLI.Support	C++/CLI support for v141 build tools (14.16)	16.3.29207.166
Microsoft.VisualStudio.Component.VC.v141.MFC	C++ MFC for v141 build tools (x86 & x64)	16.0.28625.61
Microsoft.VisualStudio.Component.VC.v141.MFC.ARM	C++ MFC for v141 build tools (ARM)	16.2.28915.88
Microsoft.VisualStudio.Component.VC.v141.MFC.ARM.Spectre	C++ MFC for v141 build tools with Spectre Mitigations (ARM)	16.0.28625.61
Microsoft.VisualStudio.Component.VC.v141.MFC.ARM64	C++ MFC for v141 build tools (ARM64)	16.0.28625.61
Microsoft.VisualStudio.Component.VC.v141.MFC.ARM64.Spectre	C++ MFC for v141 build tools with Spectre Mitigations (ARM64)	16.0.28625.61
Microsoft.VisualStudio.Component.VC.v141.MFC.Spectre	C++ MFC for v141 build tools with Spectre Mitigations (x86 & x64)	16.0.28625.61
Microsoft.VisualStudio.Component.VC.v141.x86.x64.Spectre	MSVC v141 - VS 2017 C++ x64/x86 Spectre-mitigated libs (v14.16)	16.4.29429.68
Microsoft.VisualStudio.Component.WinXP	C++ Windows XP Support for VS 2017 (v141) tools [Deprecated]	16.1.28811.260

Get support

Sometimes, things can go wrong. If your Visual Studio installation fails, see [Troubleshoot Visual Studio installation and upgrade issues](#) for step-by-step guidance.

We also offer a [live chat](#) (English only) support option for installation-related issues.

Here are a few more support options:

- Report product issues to us via the [Report a Problem](#) tool that appears both in the Visual Studio Installer and in the Visual Studio IDE.
- Suggest a feature, track product issues, and find answers in the [Visual Studio Developer Community](#).
- Use your [GitHub](#) account to talk to us and other Visual Studio developers in the [Visual Studio conversation in the Gitter community](#).

See also

- [Visual Studio workload and component IDs](#)
- [Visual Studio administrator guide](#)
- [Use command-line parameters to install Visual Studio](#)
 - [Command-line parameter examples](#)
- [Create an offline installation of Visual Studio](#)

Visual Studio Test Agent component directory

12/4/2019 • 3 minutes to read • [Edit Online](#)

The tables on this page list the IDs that you can use to install Visual Studio from the command line, or that you can specify as a dependency in a VSIX manifest. We'll add additional components when we release updates to Visual Studio.

Also note the following about the page:

- Each workload has its own section, followed by the workload ID and a table of the components that are available for the workload.
- By default, the **Required** components will be installed when you install the workload.
- If you choose to, you can also install the **Recommended** and **Optional** components.
- We've also added a section that lists the additional components that are not affiliated with any workload.

When you set dependencies in your VSIX manifest, you must specify Component IDs only. Use the tables on this page to determine the minimum component dependencies. In some scenarios, this might mean that you specify only one component from a workload. In other scenarios, it might mean that you specify multiple components from a single workload or multiple components from multiple workloads. For more information, see the [How to: Migrate extensibility projects to Visual Studio 2017](#) page.

For more information about how to use these IDs, see the [Use command-line parameters to install Visual Studio 2017](#) page. And, for a list of workload and component IDs for other products, see the [Visual Studio workload and component IDs](#) page.

Test Agent

ID: Microsoft.VisualStudio.Workload.TestAgent

Description: Supports running automated tests and load tests remotely

Components included by this workload

COMPONENT ID	NAME	VERSION	DEPENDENCY TYPE
Microsoft.VisualStudio.ComponentGroup.TestTools.TestAgent	Test Agent core features	15.0.27019.1	Required

Unaffiliated components

These are components that are not included with any workload, but may be selected as an individual component.

COMPONENT ID	NAME	VERSION
n/a	n/a	n/a

When you set dependencies in your VSIX manifest, you must specify Component IDs only. Use the tables on this page to determine the minimum component dependencies. In some scenarios, this might mean that you specify only one component from a workload. In other scenarios, it might mean that you specify multiple components from a single workload or multiple components from multiple workloads. For more information, see the [How to: Migrate extensibility projects to Visual Studio](#) page.

For more information about how to use these IDs, see the [Use command-line parameters to install Visual Studio](#) page. And, for a list of workload and component IDs for other products, see the [Visual Studio workload and component IDs](#) page.

Test Agent

ID: Microsoft.VisualStudio.Workload.TestAgent

Description: Supports running automated tests and load tests remotely

Components included by this workload

COMPONENT ID	NAME	VERSION	DEPENDENCY TYPE
Microsoft.VisualStudio.ComponentGroup.TestTools.TestAgent	Test Agent core features	16.0.28315.86	Required

Unaffiliated components

These are components that are not included with any workload, but may be selected as an individual component.

COMPONENT ID	NAME	VERSION
n/a	n/a	n/a

Get support

Sometimes, things can go wrong. If your Visual Studio installation fails, see [Troubleshoot Visual Studio installation and upgrade issues](#) for step-by-step guidance.

We also offer a [live chat](#) (English only) support option for installation-related issues.

Here are a few more support options:

- Report product issues to us via the [Report a Problem](#) tool that appears both in the Visual Studio Installer and in the Visual Studio IDE.
- Suggest a feature, track product issues, and find answers in the [Visual Studio Developer Community](#).
- Use your [GitHub](#) account to talk to us and other Visual Studio developers in the [Visual Studio conversation in the Gitter community](#).

See also

- [Visual Studio workload and component IDs](#)
- [Visual Studio administrator guide](#)
- [Use command-line parameters to install Visual Studio](#)
 - [Command-line parameter examples](#)
- [Create an offline installation of Visual Studio](#)

Visual Studio Test Controller component directory

12/4/2019 • 2 minutes to read • [Edit Online](#)

The tables on this page list the IDs that you can use to install Visual Studio from the command line, or that you can specify as a dependency in a VSIX manifest. We'll add additional components when we release updates to Visual Studio.

Also note the following about the page:

- Each workload has its own section, followed by the workload ID and a table of the components that are available for the workload.
- By default, the **Required** components will be installed when you install the workload.
- If you choose to, you can also install the **Recommended** and **Optional** components.
- We've also added a section that lists the additional components that are not affiliated with any workload.

When you set dependencies in your VSIX manifest, you must specify Component IDs only. Use the tables on this page to determine the minimum component dependencies. In some scenarios, this might mean that you specify only one component from a workload. In other scenarios, it might mean that you specify multiple components from a single workload or multiple components from multiple workloads. For more information, see the [How to: Migrate extensibility projects to Visual Studio 2017](#) page.

For more information about how to use these IDs, see the [Use command-line parameters to install Visual Studio 2017](#) page. And, for a list of workload and component IDs for other products, see the [Visual Studio workload and component IDs](#) page.

Test Controller

ID: Microsoft.VisualStudio.Workload.TestController

Description: Distribute automated tests to multiple machines

Components included by this workload

COMPONENT ID	NAME	VERSION	DEPENDENCY TYPE
Microsoft.VisualStudio.ComponentGroup.TestTools.TestController	Test Controller core features	15.6.27309.0	Required

Unaffiliated components

These are components that are not included with any workload, but may be selected as an individual component.

COMPONENT ID	NAME	VERSION
n/a	n/a	n/a

When you set dependencies in your VSIX manifest, you must specify Component IDs only. Use the tables on this page to determine the minimum component dependencies. In some scenarios, this might mean that you specify only one component from a workload. In other scenarios, it might mean that you specify multiple components from a single workload or multiple components from multiple workloads. For more information, see the [How to: Migrate extensibility projects to Visual Studio](#) page.

For more information about how to use these IDs, see the [Use command-line parameters to install Visual Studio](#) page. And, for a list of workload and component IDs for other products, see the [Visual Studio workload and component IDs](#) page.

Test Controller

ID: Microsoft.VisualStudio.Workload.TestController

Description: Distribute automated tests to multiple machines

Components included by this workload

COMPONENT ID	NAME	VERSION	DEPENDENCY TYPE
Microsoft.VisualStudio.ComponentGroup.TestTools.TestController	Test Controller core features	16.0.28315.86	Required

Unaffiliated components

These are components that are not included with any workload, but may be selected as an individual component.

COMPONENT ID	NAME	VERSION
n/a	n/a	n/a

Get support

Sometimes, things can go wrong. If your Visual Studio installation fails, see [Troubleshoot Visual Studio installation and upgrade issues](#) for step-by-step guidance.

We also offer a [live chat](#) (English only) support option for installation-related issues.

Here are a few more support options:

- Report product issues to us via the [Report a Problem](#) tool that appears both in the Visual Studio Installer and in the Visual Studio IDE.
- Suggest a feature, track product issues, and find answers in the [Visual Studio Developer Community](#).
- Use your [GitHub](#) account to talk to us and other Visual Studio developers in the [Visual Studio conversation in the Gitter community](#).

See also

- [Visual Studio workload and component IDs](#)
- [Visual Studio administrator guide](#)
- [Use command-line parameters to install Visual Studio](#)
 - [Command-line parameter examples](#)
- [Create an offline installation of Visual Studio](#)

Visual Studio Test Professional component directory

9/23/2019 • 2 minutes to read • [Edit Online](#)

The tables on this page list the IDs that you can use to install Visual Studio by using the command line or that you can specify as a dependency in a VSIX manifest. Note that we will add additional components as we release updates to Visual Studio.

Also note the following about the page:

- Each workload has its own section, followed by the workload ID and a table of the components that are available for the workload.
- By default, the **Required** components will be installed when you install the workload.
- If you choose to, you can also install the **Recommended** and **Optional** components.
- We've also added a section that lists the additional components that are not affiliated with any workload.

When you set dependencies in your VSIX manifest, you must specify Component IDs only. Use the tables on this page to determine our minimum component dependencies. In some scenarios, this might mean that you specify only one component from a workload. In other scenarios, it might mean that you specify multiple components from a single workload or multiple components from multiple workloads. For more information, see the [How to: Migrate Extensibility Projects to Visual Studio 2017](#) page.

For more information about how to use these IDs, see [Use Command-Line Parameters to Install Visual Studio 2017](#) page. And, for a list of workload and component IDs for other products, see [Visual Studio 2017 Workload and Component IDs](#) page.

Test Professional

ID: Microsoft.VisualStudio.Workload.TestProfessional

Description: Test Professional provides integrated testing tools targeted at generalist testers, which help them drive their testing needs across the entire testing lifecycle.

Components included by this workload

COMPONENT ID	NAME	VERSION	DEPENDENCY TYPE
Microsoft.VisualStudio.Component.TestTools.FeedbackClient	Microsoft Feedback Client	15.6.27406.0	Required
Microsoft.VisualStudio.Component.TestTools.MicrosoftTestManager	Microsoft Test Manager	15.6.27406.0	Required

Unaffiliated components

These are components that are not included with any workload, but may be selected as an individual component.

COMPONENT ID	NAME	VERSION
n/a	n/a	n/a

Get support

Sometimes, things can go wrong. If your Visual Studio installation fails, see [Troubleshoot Visual Studio installation and upgrade issues](#) for step-by-step guidance.

We also offer a [live chat](#) (English only) support option for installation-related issues.

Here are a few more support options:

- Report product issues to us via the [Report a Problem](#) tool that appears both in the Visual Studio Installer and in the Visual Studio IDE.
- Suggest a feature, track product issues, and find answers in the [Visual Studio Developer Community](#).
- Use your [GitHub](#) account to talk to us and other Visual Studio developers in the [Visual Studio conversation in the Gitter community](#).

See also

- [Visual Studio workload and component IDs](#)
- [Visual Studio administrator guide](#)
- [Use command-line parameters to install Visual Studio](#)
 - [Command-line parameter examples](#)
- [Create an offline installation of Visual Studio](#)

Visual Studio Feedback Client component directory

3/4/2019 • 2 minutes to read • [Edit Online](#)

The tables on this page list the IDs that you can use to install Visual Studio by using the command line or that you can specify as a dependency in a VSIX manifest. Note that we will add additional components as we release updates to Visual Studio.

Also note the following about the page:

- Each workload has its own section, followed by the workload ID and a table of the components that are available for the workload.
- By default, the **Required** components will be installed when you install the workload.
- If you choose to, you can also install the **Recommended** and **Optional** components.
- We've also added a section that lists the additional components that are not affiliated with any workload.

When you set dependencies in your VSIX manifest, you must specify Component IDs only. Use the tables on this page to determine our minimum component dependencies. In some scenarios, this might mean that you specify only one component from a workload. In other scenarios, it might mean that you specify multiple components from a single workload or multiple components from multiple workloads. For more information, see the [How to: Migrate Extensibility Projects to Visual Studio 2017](#) page.

For more information about how to use these IDs, see [Use Command-Line Parameters to Install Visual Studio 2017](#) page. And, for a list of workload and component IDs for other products, see [Visual Studio 2017 Workload and Component IDs](#) page.

Feedback Client

ID: Microsoft.VisualStudio.Workload.FeedbackClient

Description: Feedback client allows Stakeholders to provide rich feedback for Azure DevOps Services or Team Foundation Server.

Components included by this workload

COMPONENT ID	NAME	VERSION	DEPENDENCY TYPE
Microsoft.VisualStudio.Component.TestTools.FeedbackClient	Microsoft Feedback Client	15.6.27406.0	Required

Unaffiliated components

These are components that are not included with any workload, but may be selected as an individual component.

COMPONENT ID	NAME	VERSION
n/a	n/a	n/a

Get support

Sometimes, things can go wrong. If your Visual Studio installation fails, see [Troubleshoot Visual Studio installation](#)

and upgrade issues for step-by-step guidance.

We also offer a [Live chat](#) (English only) support option for installation-related issues.

Here are a few more support options:

- Report product issues to us via the [Report a Problem](#) tool that appears both in the Visual Studio Installer and in the Visual Studio IDE.
- Suggest a feature, track product issues, and find answers in the [Visual Studio Developer Community](#).
- Use your [GitHub](#) account to talk to us and other Visual Studio developers in the [Visual Studio conversation in the Gitter community](#).

See also

- [Visual Studio workload and component IDs](#)
- [Visual Studio administrator guide](#)
- [Use command-line parameters to install Visual Studio](#)
 - [Command-line parameter examples](#)
- [Create an offline installation of Visual Studio](#)

Visual Studio build numbers and release dates

12/4/2019 • 8 minutes to read • [Edit Online](#)

Visual Studio 2019

The following table lists the build numbers and release dates for Visual Studio 2019, to date.

VERSION	CHANNEL	RELEASE DATE	BUILD VERSION
16.5.0	Preview 1	December 3, 2019	16.5.29521.150
16.4.0	Release	December 3, 2019	16.4.29519.181
16.4.0	Preview 6	November 21, 2019	16.4.29519.161
16.3.10	Release	November 20, 2019	16.3.29519.87
16.4.0	Preview 5	November 14, 2019	16.4.29512.175
16.3.9	Release	November 12, 2019	16.3.29509.3
16.4.0	Preview 4	November 6, 2019	16.4.29505.145
16.3.8	Release	November 5, 2019	16.3.29503.13
16.4.0	Preview 3	November 4, 2019	16.4.29430.225
16.3.7	Release	October 29, 2019	16.3.29424.173
16.3.6	Release	October 22, 2019	16.3.29418.71
16.3.5	Release	October 15, 2019	16.3.29411.108
16.4.0	Preview 2	October 15, 2019	16.4.29411.138
16.0.9	Release	October 15, 2019	16.0.28803.598
16.3.4	Release	October 10, 2019	16.3.29409.12
16.3.3	Release	October 8, 2019	16.3.29403.142
16.3.2	Release	October 1, 2019	16.3.29326.143
16.3.1	Release	September 25, 2019	16.3.29324.140
16.4.0	Preview 1	September 23, 2019	16.4.29319.158
16.3.0	Release	September 23, 2019	16.3.29318.209

VERSION	CHANNEL	RELEASE DATE	BUILD VERSION
16.3.0	Preview 4	September 16, 2019	16.3.29311.281
16.2.5	Release	September 10, 2019	16.2.29306.81
16.0.8	Release	September 10, 2019	16.0.28803.584
16.2.4	Release	September 5, 2019	16.2.29230.47
16.3.0	Preview 3	September 4, 2019	16.3.29230.61
16.2.3	Release	August 20, 2019	16.2.29215.179
16.2.2	Release	August 13, 2019	16.2.29209.62
16.3.0	Preview 2	August 13, 2019	16.3.29209.152
16.0.7	Release	August 13, 2019	16.0.28803.571
16.2.1	Release	August 6, 2019	16.2.29201.188
16.2.0	Release	July 24, 2019	16.2.29123.88
16.3.0	Preview 1	July 24, 2019	16.3.29123.89
16.2.0	Preview 4	July 16, 2019	16.2.29111.141
16.1.6	Release	July 9, 2019	16.1.29102.190
16.0.6	Release	July 9, 2019	16.0.28803.540
16.1.5	Release	July 2, 2019	16.1.29025.244
16.1.4	Release	June 25, 2019	16.1.29020.237
16.2.0	Preview 3	June 25, 2019	16.2.29021.104
16.1.3	Release	June 11, 2019	16.1.29009.5
16.2.0	Preview 2	June 11, 2019	16.2.29006.145
16.0.5	Release	June 11, 2019	16.0.28803.514
16.1.2	Release	June 5, 2019	16.1.29001.49
16.1.1	Release	May 24, 2019	16.1.28922.388
16.1.0	Release	May 21, 2019	16.1.28917.181
16.2.0	Preview 1	May 21, 2019	16.2.28917.182

Version	Channel	Release Date	Build Version
16.0.4	Release	May 14, 2019	16.0.28803.452
16.1.0	Preview 3	May 6, 2019	16.1.28902.138
16.0.3	Release	April 30, 2019	16.0.28803.352
16.1.0	Preview 2	April 23, 2019	16.1.28822.285
16.0.2	Release	April 18, 2019	16.0.28803.202
16.1.0	Preview 1	April 10, 2019	16.1.28809.33
16.0.1	Release	April 9, 2019	16.0.28803.156
16.0.1	Preview 1	April 9, 2019	16.0.28803.156
16.0.0	Release	April 2, 2019	16.0.28729.10
16.0.0	Preview 5	April 2, 2019	16.0.28729.10
16.0.0	Release Candidate 4 (RC.4)	March 26, 2019	16.0.28721.148
16.0.0	Preview 4.4	March 26, 2019	16.0.28721.148
16.0.0	Release Candidate 3 (RC.3)	March 19, 2019	16.0.28714.193
16.0.0	Preview 4.3	March 19, 2019	16.0.28714.193
16.0.0	Release Candidate 2 (RC.2)	March 12, 2019	16.0.28711.60
16.0.0	Preview 4.2	March 12, 2019	16.0.28711.60
16.0.0	Release Candidate 1 Svc1 (RC.1 Svc1)	March 6, 2019	16.0.28705.295
16.0.0	Preview 4.1 Svc1	March 6, 2019	16.0.28705.295
16.0.0	Release Candidate 1 (RC.1)	March 5, 2019	16.0.28701.123
16.0.0	Preview 4.1	March 5, 2019	16.0.28701.123
16.0.0	Release Candidate (RC)	February 27, 2019	16.0.28625.133
16.0.0	Preview 4	February 27, 2019	16.0.28625.133
16.0.0	Preview 3	February 13, 2019	16.0.28608.199
16.0.0	Preview 2.2	February 5, 2019	16.0.28602.52
16.0.0	Preview 2.1	January 31, 2019	16.0.28529.54

VERSION	CHANNEL	RELEASE DATE	BUILD VERSION
16.0.0	Preview 2	January 23, 2019	16.0.28522.59
16.0.0	Preview 1.1	December 10, 2018	16.0.28408.50
16.0.0	Preview 1	December 4, 2018	16.0.28329.73

NOTE

For a list of the build numbers and release dates for the previous version, see [Visual Studio 2017 build numbers and release dates](#)

Visual Studio 2017

The following table lists the build numbers and release dates for Visual Studio 2017, to date.

VERSION	CHANNEL	RELEASE DATE	BUILD VERSION
15.9.17	Release	October 15, 2019	15.9.28307.905
15.9.16	Release	September 10, 2019	15.9.28307.858
15.0.27	Release	September 10, 2019	15.0.26228.98
15.9.15	Release	August 13, 2019	15.9.28307.812
15.0.26	Release	August 13, 2019	15.0.26228.96
15.9.14	Release	July 9, 2019	15.9.28307.770
15.0.25	Release	July 9, 2019	15.0.26228.92
15.9.13	Release	June 11, 2019	15.9.28307.718
15.0.24	Release	June 11, 2019	15.0.26228.88
15.9.12	Release	May 14, 2019	15.9.28307.665
15.0.23	Release	May 14, 2019	15.0.26228.85
15.9.11	Release	April 2, 2019	15.9.28307.586
15.9.10	Release	March 25, 2019	15.9.28307.557
15.9.9	Release	March 12, 2019	15.9.28307.518
15.0.22	Release	March 12, 2019	15.0.26228.76
15.9.8	Release	March 5, 2019	15.9.28307.481

Version	Channel	Release Date	Build Version
15.9.7	Release	February 12, 2019	15.9.28307.423
15.0.21	Release	February 12, 2019	15.0.26228.73
15.9.6	Release	January 24, 2019	15.9.28307.344
15.9.5	Release	January 8, 2019	15.9.28307.280
15.9.4	Release	December 11, 2018	15.9.28307.222
15.0.20	Release	December 11, 2018	15.0.26228.64
15.9.3	Release	November 28, 2018	15.9.28307.145
15.9.2	Release	November 19, 2018	15.9.28307.108
15.9.1	Release	November 15, 2018	15.9.28307.105
15.9.0	Release	November 13, 2018	15.9.28307.53
15.9.0 Preview 6	Preview	November 13, 2018	15.9.28307.53
15.9.0 Preview 5	Preview	November 6, 2018	15.9.28302.56
15.8.9	Release	November 2, 2018	15.8.28010.2050
15.8.8	Release	October 24, 2018	15.8.28010.2048
15.9.0 Preview 4	Preview	October 23, 2018	15.9.28219.56
15.8.7	Release	October 10, 2018	15.8.28010.2046
15.0.19	Release	October 10, 2018	15.0.26228.57
15.9.0 Preview 3	Preview	October 2, 2018	15.9.28128.56
15.8.6	Release	October 2, 2018	15.8.28010.2041
15.8.5	Release	September 20, 2018	15.8.28010.2036
15.9.0 Preview 2	Preview	September 11, 2018	15.9.28107.0
15.8.4	Release	September 11, 2018	15.8.28010.2026
15.8.3	Release	September 6, 2018	15.8.28010.2019
15.8.2	Release	August 28, 2018	15.8.28010.2016
15.0.18	Release	August 28, 2018	15.0.26228.52

Version	Channel	Release Date	Build Version
15.9.0 Preview 1	Preview	August 20, 2018	15.9.28016.0
15.8.1	Release	August 17, 2018	15.8.28010.2003
15.8.0	Release	August 14, 2018	15.8.28010.0
15.0.17	Release	August 14, 2018	15.0.26228.49
15.7.6	Release	August 2, 2018	15.7.27703.2047
15.0.16	Release	August 2, 2018	15.0.26228.48
15.8.0 Preview 5	Preview	July 26, 2018	15.8.27924.0
15.8.0 Preview 4	Preview	July 10, 2018	15.8.27906.1
15.7.5	Release	July 10, 2018	15.7.27703.2042
15.0.15	Release	July 10, 2018	15.0.26228.43
15.8.0 Preview 3	Preview	June 26, 2018	15.8.27825.0
15.7.4	Release	June 18, 2018	15.7.27703.2035
15.7.3	Release	May 31, 2018	15.7.27703.2026
15.0.14	Release	May 31, 2018	15.0.26228.37
15.8.0 Preview 2	Preview	May 31, 2018	15.8.27729.1
15.7.2	Release	May 21, 2018	15.7.27703.2018
15.8.0 Preview 1	Preview	May 8, 2018	15.8.27705.2000
15.7.1	Release	May 8, 2018	15.7.27703.2000
15.8.0 Preview 1	Preview	May 7, 2018	15.8.27705.0
15.7.0	Release	May 7, 2018	15.7.27703.1
15.7.0 Preview 6	Preview	May 3, 2018	15.7.27701.1
15.7.0 Preview 5	Preview	April 26, 2018	15.7.27625.0
15.6.7	Release	April 26, 2018	15.6.27428.2043
15.0.13	Release	April 26, 2018	15.0.26228.31
15.7.0 Preview 4	Preview	April 18, 2018	15.7.27617.1

Version	Channel	Release Date	Build Version
15.6.6	Release	April 10, 2018	15.6.27428.2037
15.0.12	Release	April 10, 2018	15.0.26228.30
15.7.0 Preview 3	Preview	April 9, 2018	15.7.27604.0
15.6.5	Release	April 4, 2018	15.6.27428.2027
15.6.4	Release	March 22, 2018	15.6.27428.2015
15.7.0 Preview 2	Preview	March 21, 2018	15.7.27520.0
15.6.3	Release	March 19, 2018	15.6.27428.2011
15.7.0 Preview 1	Preview	March 13, 2018	15.7.27512.0
15.6.2	Release	March 13, 2018	15.6.27428.2005
15.0.11	Release	March 13, 2018	15.0.26228.29
15.6.1	Release	March 8, 2018	15.6.27428.2002
15.6.1 Preview 1	Preview	March 8, 2018	15.6.27428.2002
15.6.0	Release	March 5, 2018	15.6.27428.1
15.6.0 Preview 7	Preview	March 2, 2018	15.6.27428.1
15.6.0 Preview 6	Preview	February 23, 2018	15.6.27421.1
15.0.10	Release	February 21, 2018	15.0.26228.28
15.5.7	Release	February 20, 2018	15.0.27130.2036
15.6.0 Preview 5	Preview	February 14, 2018	15.6.27413.0
15.6.0 Preview 4	Preview	February 7, 2018	15.6.27406.0
15.0.9	Release	February 2, 2018	15.0.26228.23
15.5.6	Release	January 29, 2018	15.0.27130.2027
15.5.5	Release	January 25, 2018	15.0.27130.2026
15.6.0 Preview 3	Preview	January 25, 2018	15.6.27323.2
15.5.4	Release	January 16, 2018	15.0.27130.2024
15.6.0 Preview 2	Preview	January 10, 2018	15.6.27309.0

VERSION	CHANNEL	RELEASE DATE	BUILD VERSION
15.5.3	Release	January 9, 2018	15.0.27130.2020
15.0.8	Release	January 9, 2018	15.0.26228.21
15.5.2	Release	December 14, 2017	15.0.27130.2010
15.6.0 Preview 1	Preview	December 14, 2017	15.6.27205.2004
15.5.1	Release	December 7, 2017	15.0.27130.2003
15.6.0 Preview 1	Preview	December 7, 2017	15.6.27205.0
15.0.7	Release	December 6, 2017	15.0.26228.18
15.5.0	Release	December 4, 2017	15.0.27130.0
15.5.0 Preview 5	Preview	November 30, 2017	15.0.27128.1
15.4.5	Release	November 27, 2017	15.0.27004.2010
15.5.0 Preview 4	Preview	November 14, 2017	15.0.27110.0
15.4.4	Release	November 14, 2017	15.0.27004.2009
15.0.6	Release	November 14, 2017	15.0.26228.17
15.4.3	Release	November 8, 2017	15.0.27004.2008
15.5.0 Preview 3	Preview	November 6, 2017	15.0.27102.0
15.4.2	Release	October 31, 2017	15.0.27004.2006
15.5.0 Preview 2	Preview	October 23, 2017	15.0.27019.1
15.4.1	Release	October 19, 2017	15.0.27004.2005
15.5 Preview 1	Preview	October 11, 2017	15.0.27009.1
15.4.0	Release	October 9, 2017	15.0.27004.2002
15.4 Preview 6	Preview	October 9, 2017	15.0.27004.20002
15.4 Preview 5	Preview	October 6, 2017	15.0.27004.2000
15.4 Preview 4	Preview	October 2, 2017	15.0.26929.2
15.4 Preview 3	Preview	September 21, 2017	15.0.26923.00
15.3.5	Release	September 19, 2017	15.0.26730.16

Version	Channel	Release Date	Build Version
15.0.5	Release	September 18, 2017	15.0.26228.16
15.3.4	Release	September 12, 2017	15.0.26730.15
15.4 Preview 2	Preview	September 11, 2017	15.0.26906.1
15.3.3	Release	August 29, 2017	15.0.26730.12
15.4 Preview 1	Preview	August 24, 2017	15.0.26823.01
15.3.2	Release	August 22, 2017	15.0.26730.10
15.3.1	Release	August 18, 2017	15.0.26730.08
15.3.1 Preview 1	Preview	August 18, 2017	15.0.26730.08
15.3.1	Release	August 18, 2017	15.0.26730.08
15.4 Preview 1	Preview	August 24, 2017	15.0.26823.1
15.3.0	Release	August 14, 2017	15.0.26730.3
15.3 Preview 7.1	Preview	August 11, 2017	15.0.26730.3
15.3 Preview 7	Preview	August 1, 2017	15.0.26730.0
15.3 Preview 6	Preview	July 26, 2017	15.0.26724.1
15.3 Preview 5	Preview	July 24, 2017	15.0.26720.02
15.2.6	Release	July 17, 2017	15.0.26430.16
15.3 Preview 4	Preview	July 12, 2017	15.0.26711.1
15.2.5	Release	July 6, 2017	15.0.26430.15
15.3 Preview 3	Preview	June 26, 2017	15.0.26621.2
15.2.4	Release	June 21, 2017	15.0.26430.14
15.3 Preview 2.1	Preview	June 20, 2017	15.0.26608.5
15.2.3	Release	June 9, 2017	15.0.26430.13
15.3 Preview 2	Preview	June 8, 2017	15.0.26606.0
15.2.2	Release	May 30, 2017	15.0.26430.12
15.0.4	Release	May 23, 2017	15.0.26228.13

Version	Channel	Release Date	Build Version
15.2.1	Release	May 12, 2017	15.0.26430.6
15.3 Preview 1.1	Preview	May 11, 2017	15.0.26510.0
15.3 Preview 1	Preview	May 10, 2017	15.0.26507.0
15.2.0	Release	May 10, 2017	15.0.26430.4
15.2 Preview 4	Preview	May 3, 2017	15.0.26430.1
15.2 Preview 3	Preview	April 26, 2017	15.0.26424.2
15.2 Preview 2	Preview	April 20, 2017	15.0.26419.1
15.2 Preview 1	Preview	April 17, 2017	15.0.26412.1
15.1.2	Release	April 17, 2017	15.0.26403.7
15.1.1	Release	April 10, 2017	15.0.26403.3
15.1.0	Release	April 5, 2017	15.0.26403.0
15.0.3	Release	March 31, 2017	15.0.26228.12
15.0.2	Release	March 28, 2017	15.0.26228.10
15.1 Preview 3	Preview	March 27, 2017	15.0.26323.1
15.1 Preview 2	Preview	March 16, 2017	15.0.26315.0
15.0.1	Release	March 14, 2017	15.0.26228.9
15.1 Preview 1	Preview	March 7, 2017	15.0.26304.0
15.0.0	Release	March 7, 2017	15.0.26228.4

NOTE

For more information about build numbers and release dates for the next version of Visual Studio, see the [Visual Studio 2019 build numbers and release dates](#) page.

Get support

Sometimes, things can go wrong. If your Visual Studio installation fails, see [Troubleshoot Visual Studio installation and upgrade issues](#) for step-by-step guidance.

We also offer a [live chat](#) (English only) support option for installation-related issues.

Here are a few more support options:

- Report product issues to us via the [Report a Problem](#) tool that appears both in the Visual Studio Installer and in

the Visual Studio IDE.

- Suggest a feature, track product issues, and find answers in the [Visual Studio Developer Community](#).
- Use your [GitHub](#) account to talk to us and other Visual Studio developers in the [Visual Studio conversation in the Gitter community](#).

See also

- [Visual Studio release rhythm](#)
- [Visual Studio administrator guide](#)
- [Use command-line parameters to install Visual Studio](#)
- [Tools for detecting and managing Visual Studio instances](#)

Visual Studio images on Azure

12/7/2019 • 4 minutes to read • [Edit Online](#)

Using Visual Studio in a preconfigured Azure virtual machine (VM) is a quick, easy way to go from nothing to an up-and-running development environment. System images with different Visual Studio configurations are available in the [Azure Marketplace](#).

New to Azure? [Create a free Azure account](#).

What configurations and versions are available?

Images for the most recent major versions, Visual Studio 2019, Visual Studio 2017 and Visual Studio 2015, can be found in the Azure Marketplace. For each released major version, you see the originally "released to web" (RTW) version and the latest updated versions. Each of these versions offers the Visual Studio Enterprise and the Visual Studio Community editions. These images are updated at least every month to include the latest Visual Studio and Windows updates. While the names of the images remain the same, each image's description includes the installed product version and the image's "as of" date.

RELEASE VERSION	EDITIONS	PRODUCT VERSION
Visual Studio 2019: Latest (Version 16.4)	Enterprise, Community	Version 16.4.0
Visual Studio 2019: RTW	Enterprise	Version 16.0.9
Visual Studio 2017: Latest (Version 15.9)	Enterprise, Community	Version 15.9.17
Visual Studio 2017: RTW	Enterprise, Community	Version 15.0.27
Visual Studio 2015: Latest (Update 3)	Enterprise, Community	Version 14.0.25431.01

NOTE

In accordance with Microsoft servicing policy, the originally released (RTW) version of Visual Studio 2015 has expired for servicing. Visual Studio 2015 Update 3 is the only remaining version offered for the Visual Studio 2015 product line.

For more information, see the [Visual Studio Servicing Policy](#).

What features are installed?

Each image contains the recommended feature set for that Visual Studio edition. Generally, the installation includes:

- All available workloads, including each workload's recommended optional components
- .NET 4.6.2 and .NET 4.7 SDKs, Targeting Packs, and Developer Tools
- Visual F#
- GitHub Extension for Visual Studio
- LINQ to SQL Tools

We use the following command line to install Visual Studio when building the images:

```
vs_enterprise.exe --allWorkloads --includeRecommended --passive ^
--add Microsoft.Net.Component.4.7.SDK ^
--add Microsoft.Net.Component.4.7.TargetingPack ^
--add Microsoft.Net.Component.4.6.2.SDK ^
--add Microsoft.Net.Component.4.6.2.TargetingPack ^
--add Microsoft.Net.ComponentGroup.4.7.DeveloperTools ^
--add Microsoft.VisualStudio.Component.FSharp ^
--add Component.GitHub.VisualStudio ^
--add Microsoft.VisualStudio.Component.LinqToSql
```

If the images don't include a Visual Studio feature that you require, provide feedback through the feedback tool in the upper-right corner of the page.

What size VM should I choose?

Azure offers a full range of virtual machine sizes. Because Visual Studio is a powerful, multi-threaded application, you want a VM size that includes at least two processors and 7 GB of memory. We recommend the following VM sizes for the Visual Studio images:

- Standard_D2_v3
- Standard_D2s_v3
- Standard_D4_v3
- Standard_D4s_v3
- Standard_D2_v2
- Standard_D2S_v2
- Standard_D3_v2

For more information on the latest machine sizes, see [Sizes for Windows virtual machines in Azure](#).

With Azure, you can rebalance your initial choice by resizing the VM. You can either provision a new VM with a more appropriate size, or resize your existing VM to different underlying hardware. For more information, see [Resize a Windows VM](#).

After the VM is running, what's next?

Visual Studio follows the "bring your own license" model in Azure. As with an installation on proprietary hardware, one of the first steps is licensing your Visual Studio installation. To unlock Visual Studio, either:

- Sign in with a Microsoft account that's associated with a Visual Studio subscription
- Unlock Visual Studio with the product key that came with your initial purchase

For more information, see [Sign in to Visual Studio](#) and [How to unlock Visual Studio](#).

How do I save the development VM for future or team use?

The spectrum of development environments is huge, and there's real cost associated with building out the more complex environments. Regardless of your environment's configuration, you can save, or capture, your configured VM as a "base image" for future use or for other members of your team. Then, when booting a new VM, you provision it from the base image rather than the Azure Marketplace image.

A quick summary: Use the System Preparation tool (Sysprep) and shut down the running VM, and then capture (*Figure 1*) the VM as an image through the UI in the Azure portal. Azure saves the `.vhdx` file that contains the image in the storage account of your choosing. The new image then shows up as an Image resource in your subscription's list of resources.



(Figure 1) Capture an image through the Azure portal's UI.

For more information, see [Create a managed image of a generalized VM in Azure](#).

IMPORTANT

Don't forget to use Sysprep to prepare the VM. If you miss that step, Azure can't provision a VM from the image.

NOTE

You still incur some cost for storage of the images, but that incremental cost can be insignificant compared to the overhead costs to rebuild the VM from scratch for each team member who needs one. For instance, it costs a few dollars to create and store a 127-GB image for a month that's reusable by your entire team. However, these costs are insignificant compared to hours each employee invests to build out and validate a properly configured dev box for their individual use.

Additionally, your development tasks or technologies might need more scale, like varieties of development configurations and multiple machine configurations. You can use Azure DevTest Labs to create *recipes* that automate construction of your "golden image." You can also use DevTest Labs to manage policies for your team's running VMs. [Using Azure DevTest Labs for developers](#) is the best source for more information on DevTest Labs.

Next steps

Now that you know about the preconfigured Visual Studio images, the next step is to create a new VM:

- [Create a VM through the Azure portal](#)
- [Windows Virtual Machines overview](#)

Install Build Tools into a container

11/27/2019 • 6 minutes to read • [Edit Online](#)

You can install Visual Studio Build Tools into a Windows container to support continuous integration and continuous delivery (CI/CD) workflows. This article guides you through what Docker configuration changes are required as well as what [workloads and components](#) you can install in a container.

[Containers](#) are a great way to package a consistent build system you can use not only in a CI/CD server environment but for development environments as well. For example, you can mount your source code into a container to be built by a customized environment while you continue to use Visual Studio or other tools to write your code. If your CI/CD workflow uses the same container image, you can rest assured that your code builds consistently. You can use containers for runtime consistency as well, which is common for micro-services using multiple containers with an orchestration system; however, is beyond the scope of this article.

If Visual Studio Build Tools does not have what you require to build your source code, these same steps can be used for other Visual Studio products. Do note, however, that Windows containers do not support an interactive user interface so all commands must be automated.

Before you begin

Some familiarity with [Docker](#) is assumed below. If you're not already familiar with running Docker on Windows, read about how to [install and configure the Docker engine on Windows](#).

The base image below is a sample and may not work for your system. Read [Windows container version compatibility](#) to determine which base image you should use for your environment.

Create and build the Dockerfile

Save the following example Dockerfile to a new file on your disk. If the file is named simply "Dockerfile", it is recognized by default.

WARNING

This example Dockerfile excludes only earlier Windows SDKs that cannot be installed into containers. Earlier releases cause the build command to fail.

1. Open a command prompt.
2. Create a new directory (recommended):

```
mkdir C:\BuildTools
```

3. Change directories to this new directory:

```
cd C:\BuildTools
```

4. Save the following content to C:\BuildTools\Dockerfile.

```

# escape=`

# Use the latest Windows Server Core image with .NET Framework 4.7.2.
FROM mcr.microsoft.com/dotnet/framework/sdk:4.7.2-windowsservercore-ltsc2019

# Restore the default Windows shell for correct batch processing.
SHELL ["cmd", "/S", "/C"]

# Download the Build Tools bootstrapper.
ADD https://aka.ms/vs/15/release/vs_buildtools.exe C:\TEMP\vs_buildtools.exe

# Install Build Tools excluding workloads and components with known issues.
RUN C:\TEMP\vs_buildtools.exe --quiet --wait --norestart --nocache `

    --installPath C:\BuildTools `

    --all `

    --remove Microsoft.VisualStudio.Component.Windows10SDK.10240 `

    --remove Microsoft.VisualStudio.Component.Windows10SDK.10586 `

    --remove Microsoft.VisualStudio.Component.Windows10SDK.14393 `

    --remove Microsoft.VisualStudio.Component.Windows81SDK `

|| IF "%ERRORLEVEL%"=="3010" EXIT 0

# Start developer command prompt with any other commands specified.
ENTRYPOINT C:\BuildTools\Common7\Tools\VsDevCmd.bat &&

# Default to PowerShell if no other command specified.
CMD ["powershell.exe", "-NoLogo", "-ExecutionPolicy", "Bypass"]

```

WARNING

If you base your image directly on `microsoft/windowsservercore` or `mcr.microsoft.com/windows/servercore` (see [Microsoft syndicates container catalog](#)), the .NET Framework might not install properly and no install error is indicated. Managed code might not run after the install is complete. Instead, base your image on [microsoft/dotnet-framework:4.7.2](#) or later. Also note that images that are tagged version 4.7.2 or later might use PowerShell as the default `SHELL`, which will cause the `RUN` and `ENTRYPOINT` instructions to fail.

Visual Studio 2017 version 15.8 or earlier (any product) will not properly install on `mcr.microsoft.com/windows/servercore:1809` or later. No error is displayed.

See [Windows container version compatibility](#) to see which container OS versions are supported on which host OS versions, and [Known issues for containers](#) for known issues.

```

# escape=`

# Use the latest Windows Server Core image with .NET Framework 4.8.
FROM mcr.microsoft.com/dotnet/framework/sdk:4.8-windowsservercore-ltsc2019

# Restore the default Windows shell for correct batch processing.
SHELL ["cmd", "/S", "/C"]

# Download the Build Tools bootstrapper.
ADD https://aka.ms/vs/16/release/vs_buildtools.exe C:\TEMP\vs_buildtools.exe

# Install Build Tools excluding workloads and components with known issues.
RUN C:\TEMP\vs_buildtools.exe --quiet --wait --norestart --nocache `

    --installPath C:\BuildTools `

    --all `

    --remove Microsoft.VisualStudio.Component.Windows10SDK.10240 `

    --remove Microsoft.VisualStudio.Component.Windows10SDK.10586 `

    --remove Microsoft.VisualStudio.Component.Windows10SDK.14393 `

    --remove Microsoft.VisualStudio.Component.Windows81SDK `

|| IF "%ERRORLEVEL%"=="3010" EXIT 0

# Start developer command prompt with any other commands specified.
ENTRYPOINT C:\BuildTools\Common7\Tools\VsDevCmd.bat &&

# Default to PowerShell if no other command specified.
CMD ["powershell.exe", "-NoLogo", "-ExecutionPolicy", "Bypass"]

```

WARNING

If you base your image directly on `microsoft/windowsservercore`, the .NET Framework might not install properly and no install error is indicated. Managed code might not run after the install is complete. Instead, base your image on `microsoft/dotnet-framework:4.8` or later. Also note that images that are tagged version 4.8 or later might use PowerShell as the default `SHELL`, which will cause the `RUN` and `ENTRYPOINT` instructions to fail.

See [Windows container version compatibility](#) to see which container OS versions are supported on which host OS versions, and [Known issues for containers](#) for known issues.

NOTE

Error code `3010` is used to indicate success with a reboot required, see [MsiExec.exe error messages](#) for more information.

- Run the following command within that directory.

```
docker build -t buildtools2017:latest -m 2GB .
```

This command builds the Dockerfile in the current directory using 2 GB of memory. The default 1 GB is not sufficient when some workloads are installed; however, you might be able to build with only 1 GB of memory depending on your build requirements.

The final image is tagged "buildtools2017:latest" so you can easily run it in a container as "buildtools2017" since the "latest" tag is the default if no tag is specified. If you want to use a specific version of Visual Studio Build Tools 2017 in a more [advanced scenario](#), you might instead tag the container with a specific Visual Studio build number as well as "latest" so containers can use a specific version consistently.

```
docker build -t buildtools2019:latest -m 2GB .
```

This command builds the Dockerfile in the current directory using 2 GB of memory. The default 1 GB is not sufficient when some workloads are installed; however, you might be able to build with only 1 GB of memory depending on your build requirements.

The final image is tagged "buildtools2019:latest" so you can easily run it in a container as "buildtools2019" since the "latest" tag is the default if no tag is specified. If you want to use a specific version of Visual Studio Build Tools 2019 in a more [advanced scenario](#), you might instead tag the container with a specific Visual Studio build number as well as "latest" so containers can use a specific version consistently.

Using the built image

Now that you have created an image, you can run it in a container to do both interactive and automated builds. The example uses the Developer Command Prompt, so your PATH and other environment variables are already configured.

1. Open a command prompt.
2. Run the container to start a PowerShell environment with all developer environment variables set:

```
docker run -it buildtools2017
```

```
docker run -it buildtools2019
```

To use this image for your CI/CD workflow, you can publish it to your own [Azure Container Registry](#) or other internal [Docker registry](#) so servers need only to pull it.

Get support

Sometimes, things can go wrong. If your Visual Studio installation fails, see [Troubleshoot Visual Studio installation and upgrade issues](#) for step-by-step guidance.

We also offer a [Live chat](#) (English only) support option for installation-related issues.

Here are a few more support options:

- Report product issues to us via the [Report a Problem](#) tool that appears both in the Visual Studio Installer and in the Visual Studio IDE.
- Suggest a feature, track product issues, and find answers in the [Visual Studio Developer Community](#).
- Use your [GitHub](#) account to talk to us and other Visual Studio developers in the [Visual Studio conversation in the Gitter community](#).

See also

- [Advanced Example for Containers](#)
- [Known Issues for Containers](#)
- [Visual Studio Build Tools workload and component IDs](#)

Advanced example for containers

7/11/2019 • 4 minutes to read • [Edit Online](#)

The sample Dockerfile in [Install Build Tools into a container](#) always uses the [microsoft/dotnet-framework:4.7.2](#) image based on the latest microsoft/windowsservercore image and the latest Visual Studio Build Tools installer. If you publish this image to a [Docker registry](#) for others to pull, this image might be okay for many scenarios. However, in practice it's more common to be specific about what base image you use, what binaries you download, and which tool versions you install.

The sample Dockerfile in [Install Build Tools into a container](#) always uses the [microsoft/dotnet-framework:4.8](#) image based on the latest microsoft/windowsservercore image and the latest Visual Studio Build Tools installer. If you publish this image to a [Docker registry](#) for others to pull, this image might be okay for many scenarios. However, in practice it's more common to be specific about what base image you use, what binaries you download, and which tool versions you install.

The following example Dockerfile uses a specific version tag of the microsoft/dotnet-framework image. Using a specific tag for a base image is commonplace and makes it easy to remember that building or rebuilding images always has the same basis.

NOTE

You cannot install Visual Studio into `microsoft/windowsservercore:10.0.14393.1593` or any image based on it, which has known issues launching the installer in a container. For more information, see [Known issues for containers](#).

The following example downloads the latest release of Build Tools. If you want to use an earlier version of Build Tools that you can install into a container later, you must first [create](#) and [maintain](#) a layout.

Install script

To collect logs when an install error occurs, create a batch script that's named "Install.cmd" in the working directory that includes the following content:

```
@if not defined _echo echo off
setlocal enabledelayedexpansion

call %*
if "%ERRORLEVEL%"=="3010" (
    exit /b 0
) else (
    if not "%ERRORLEVEL%"=="0" (
        set ERR=%ERRORLEVEL%
        call C:\TEMP\collect.exe -zip:C:\vslogs.zip

        exit /b !ERR!
    )
)
```

Dockerfile

In the working directory, create the "Dockerfile" with the following content:

```

# escape=`

# Use a specific tagged image. Tags can be changed, though that is unlikely for most images.
# You could also use the immutable tag
@sha256:3eaa3ba18f45e6561f32d8dd927045413f1dd043d7d29fb581f5cb3c6f7d7481
ARG FROM_IMAGE=mcr.microsoft.com/dotnet/framework/sdk:4.7.2-windowsservercore-ltsc2019
FROM ${FROM_IMAGE}

# Copy our Install script.
COPY Install.cmd C:\TEMP\

# Download collect.exe in case of an install failure.
ADD https://aka.ms/vscollect.exe C:\TEMP\collect.exe

# Use the latest release channel. For more control, specify the location of an internal layout.
ARG CHANNEL_URL=https://aka.ms/vs/15/release/channel
ADD ${CHANNEL_URL} C:\TEMP\VisualStudio.chman

# Download and install Build Tools excluding workloads and components with known issues.
ADD https://aka.ms/vs/15/release/vs_buildtools.exe C:\TEMP\vs_buildtools.exe
RUN C:\TEMP\Install.cmd C:\TEMP\vs_buildtools.exe --quiet --wait --norestart --nocache `

    --installPath C:\BuildTools `

    --channelUri C:\TEMP\VisualStudio.chman `

    --installChannelUri C:\TEMP\VisualStudio.chman `

    --all `

    --remove Microsoft.VisualStudio.Component.Windows10SDK.10240 `

    --remove Microsoft.VisualStudio.Component.Windows10SDK.10586 `

    --remove Microsoft.VisualStudio.Component.Windows10SDK.14393 `

    --remove Microsoft.VisualStudio.Component.Windows81SDK

# Start developer command prompt with any other commands specified.
ENTRYPOINT C:\BuildTools\Common7\Tools\VsDevCmd.bat &&

# Default to PowerShell if no other command specified.
CMD ["powershell.exe", "-NoLogo", "-ExecutionPolicy", "Bypass"]

```

WARNING

Visual Studio 2017 version 15.8 or earlier (any product) will not properly install on mcr.microsoft.com/windows/servercore:1809 or later. No error is displayed.

See [Known issues for containers](#) for more information.

```

# escape=`

# Use a specific tagged image. Tags can be changed, though that is unlikely for most images.
# You could also use the immutable tag
@sha256:324e9ab7262331ebb16a4100d0fb1cfb804395a766e3bb1806c62989d1fc1326
ARG FROM_IMAGE=mcr.microsoft.com/dotnet/framework/sdk:4.8-windowsservercore-ltsc2019
FROM ${FROM_IMAGE}

# Copy our Install script.
COPY Install.cmd C:\TEMP\

# Download collect.exe in case of an install failure.
ADD https://aka.ms/vscollect.exe C:\TEMP\collect.exe

# Use the latest release channel. For more control, specify the location of an internal layout.
ARG CHANNEL_URL=https://aka.ms/vs/16/release/channel
ADD ${CHANNEL_URL} C:\TEMP\VisualStudio.chman

# Download and install Build Tools excluding workloads and components with known issues.
ADD https://aka.ms/vs/16/release/vs_buildtools.exe C:\TEMP\vs_buildtools.exe
RUN C:\TEMP\Install.cmd C:\TEMP\vs_buildtools.exe --quiet --wait --norestart --nocache `

    --installPath C:\BuildTools `

    --channelUri C:\TEMP\VisualStudio.chman `

    --installChannelUri C:\TEMP\VisualStudio.chman `

    --all `

    --remove Microsoft.VisualStudio.Component.Windows10SDK.10240 `

    --remove Microsoft.VisualStudio.Component.Windows10SDK.10586 `

    --remove Microsoft.VisualStudio.Component.Windows10SDK.14393 `

    --remove Microsoft.VisualStudio.Component.Windows81SDK

# Start developer command prompt with any other commands specified.
ENTRYPOINT C:\BuildTools\Common7\Tools\VsDevCmd.bat &&

# Default to PowerShell if no other command specified.
CMD ["powershell.exe", "-NoLogo", "-ExecutionPolicy", "Bypass"]

```

Run the following command to build the image in the current working directory:

```
docker build -t buildtools2017:15.6.27428.2037 -t buildtools2017:latest -m 2GB .
```

```
docker build -t buildtools2019:16.0.28714.193 -t buildtools2019:latest -m 2GB .
```

Optionally pass either or both `FROM_IMAGE` or `CHANNEL_URL` arguments using the `--build-arg` command-line switch to specify a different base image or the location of an internal layout to maintain a fixed image.

Diagnosing install failures

This example downloads specific tools and validates that the hashes match. It also downloads the latest Visual Studio and .NET log collection utility so that if an install failure does occur, you can copy the logs to your host machine to analyze the failure.

```
> docker build -t buildtools2017:15.6.27428.2037 -t buildtools2017:latest -m 2GB .
Sending build context to Docker daemon

...
Step 8/10 : RUN C:\TEMP\Install.cmd C:\TEMP\vs_buildtools.exe --quiet --wait --norestart --nocache ...
--> Running in 4b62b4ce3a3c
The command 'cmd /S /C C:\TEMP\Install.cmd C:\TEMP\vs_buildtools.exe ...' returned a non-zero code: 1603

> docker cp 4b62b4ce3a3c:C:\vslogs.zip "%TEMP%\vslogs.zip"
```

```
> docker build -t buildtools2019:16.0.28714.193 -t buildtools2019:latest -m 2GB .
Sending build context to Docker daemon

...
Step 8/10 : RUN C:\TEMP\Install.cmd C:\TEMP\vs_buildtools.exe --quiet --wait --norestart --nocache ...
--> Running in 4b62b4ce3a3c
The command 'cmd /S /C C:\TEMP\Install.cmd C:\TEMP\vs_buildtools.exe ...' returned a non-zero code: 1603

> docker cp 4b62b4ce3a3c:C:\vslogs.zip "%TEMP%\vslogs.zip"
```

After the last line finishes executing, open "%TEMP%\vslogs.zip" on your machine, or submit an issue on the [Developer Community](#) website.

Get support

Sometimes, things can go wrong. If your Visual Studio installation fails, see [Troubleshoot Visual Studio installation and upgrade issues](#) for step-by-step guidance.

We also offer a [live chat](#) (English only) support option for installation-related issues.

Here are a few more support options:

- Report product issues to us via the [Report a Problem](#) tool that appears both in the Visual Studio Installer and in the Visual Studio IDE.
- Suggest a feature, track product issues, and find answers in the [Visual Studio Developer Community](#).
- Use your [GitHub](#) account to talk to us and other Visual Studio developers in the [Visual Studio conversation in the Gitter community](#).

See also

- [Install Build Tools into a Container](#)
- [Known Issues for Containers](#)
- [Visual Studio Build Tools workload and component IDs](#)

Known issues for containers

10/24/2019 • 2 minutes to read • [Edit Online](#)

There are a few issues when installing Visual Studio into a Docker container.

Windows container

The following known issues occur when you install Visual Studio Build Tools into a Windows container.

- You cannot install Visual Studio into a container based on image `microsoft/windowsservercore:10.0.14393.1593`. Images tagged with Windows versions before or after 10.0.14393 should work.
- You cannot install Windows SDK version 10.0.14393 or earlier. Certain packages fail to install and workloads that depend on those packages will not work.
- Pass `-m 2GB` (or more) when building the image. Some workloads require more memory than the default 1 GB when installed.
- Configure Docker to use disks larger than the default 20 GB.
- Pass `--norestart` on the command line. As of this writing, attempting to restart a Windows container from within the container returns `ERROR_TOO_MANY_OPEN_FILES` to the host.
- If you base your image directly on `microsoft/windowsservercore`, the .NET Framework might not install properly and no install error is indicated. Managed code might not run after the install is complete. Instead, base your image on [microsoft/dotnet-framework:4.7.1](#) or later. As an example, you might see an error when building with MSBuild that's similar to the following:

```
C:\BuildTools\MSBuild\15.0\bin\Roslyn\Microsoft.CSharp.Core.targets(84,5): error MSB6003: The  
specified task executable "csc.exe" could not be run. Could not load file or assembly  
'System.IO.FileSystem, Version=4.0.1.0, Culture=neutral, PublicKeyToken=b03f5f7f11d50a3a' or one  
of its dependencies. The system cannot find the file specified.
```

- You cannot install Visual Studio 2017 version 15.8 or earlier (any product) on `mcr.microsoft.com/windows/servercore:1809` or later. See <https://aka.ms/setup/containers/servercore1809> for more information.

Build Tools container

The following known issues might occur when you use a Build Tools container. To see whether issues have been fixed or if there are other known issues, visit <https://developercommunity.visualstudio.com>.

- IntelliTrace might not work in [some scenarios](#) within a container.
- On older versions of Docker for Windows, the default container image size is only 20 GB and will not fit Build Tools. Follow [instructions to change image size](#) to 127 GB or more.

Get support

Sometimes, things can go wrong. If your Visual Studio installation fails, see [Troubleshoot Visual Studio installation and upgrade issues](#) for step-by-step guidance.

We also offer a [live chat](#) (English only) support option for installation-related issues.

Here are a few more support options:

- Report product issues to us via the [Report a Problem](#) tool that appears both in the Visual Studio Installer and in the Visual Studio IDE.
- Suggest a feature, track product issues, and find answers in the [Visual Studio Developer Community](#).
- Use your [GitHub](#) account to talk to us and other Visual Studio developers in the [Visual Studio conversation in the Gitter community](#).

See also

- [Install Build Tools into a Container](#)
- [Advanced Example for Containers](#)
- [Visual Studio Build Tools workload and component IDs](#)

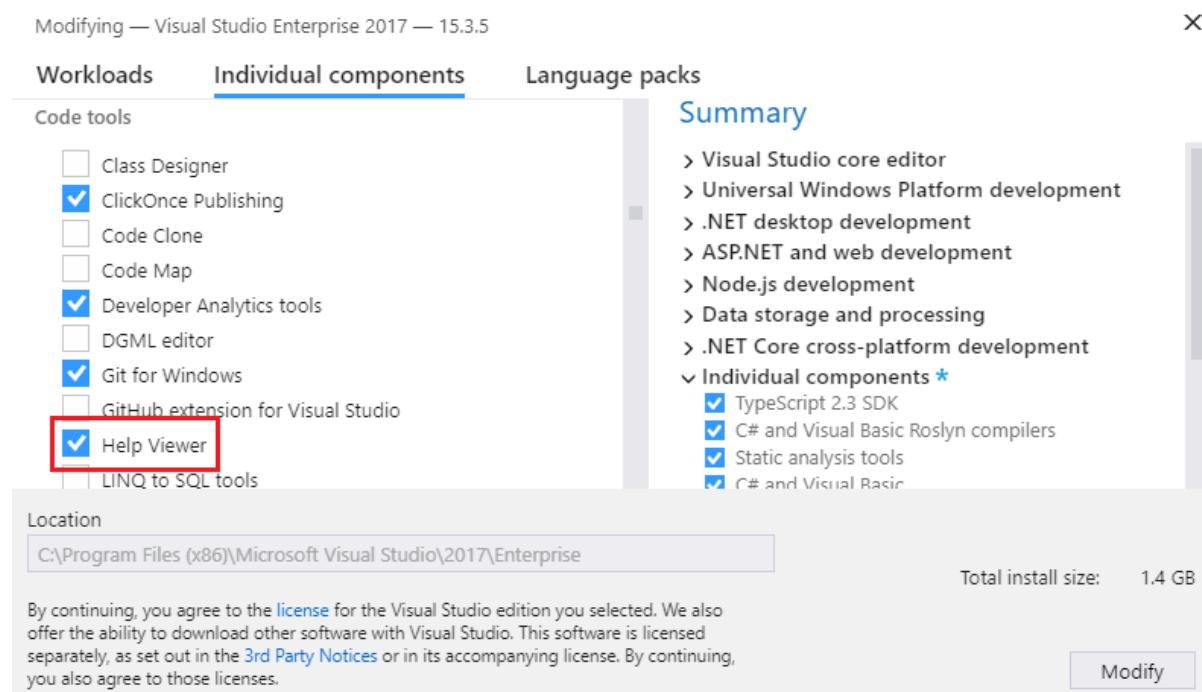
Microsoft Help Viewer installation

7/11/2019 • 2 minutes to read • [Edit Online](#)

Several products can display Help content in Microsoft Help Viewer, including Visual Studio and SQL Server.

Help Viewer is an optional installation component of Visual Studio. To install it through Visual Studio Installer, follow these steps:

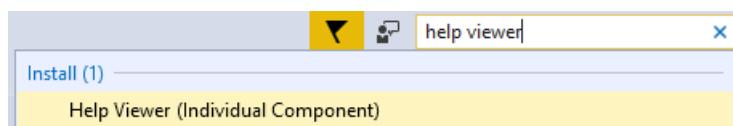
1. Open **Visual Studio Installer** from the Start menu or, if you have Visual Studio open, you can choose **Tools > Get Tools and Features** to open Visual Studio Installer.
2. Choose the **Individual Components** tab, then select **Help Viewer** under the **Code tools** section.



3. Choose the **Modify** button to start the installation of Microsoft Help Viewer.

Another way to easily install Microsoft Help Viewer is through the search box:

1. Press **Ctrl+Q** and then type or enter **help viewer** in the search box.



2. Choose the result called **Help Viewer (Individual Component)**.
3. In the dialog box that opens, choose the **Install** button.

The workloads and components that you selected will be installed. We might need to restart Visual Studio or your computer during the installation.

Individual components

Help Viewer

Total install size: 1.4 GB

[Customize](#)

By continuing, you agree to the [license](#) for the Visual Studio edition you selected. We also offer the ability to download other software with Visual Studio. This software is licensed separately, as set out in the [3rd Party Notices](#) or in its accompanying license. By continuing, you also agree to those licenses.

[Install](#)

[Cancel](#)

See also

- [Microsoft Help Viewer](#)
- [Help viewer and offline content for SQL Server](#)

Quickstart: First look at the Visual Studio IDE

10/18/2019 • 5 minutes to read • [Edit Online](#)

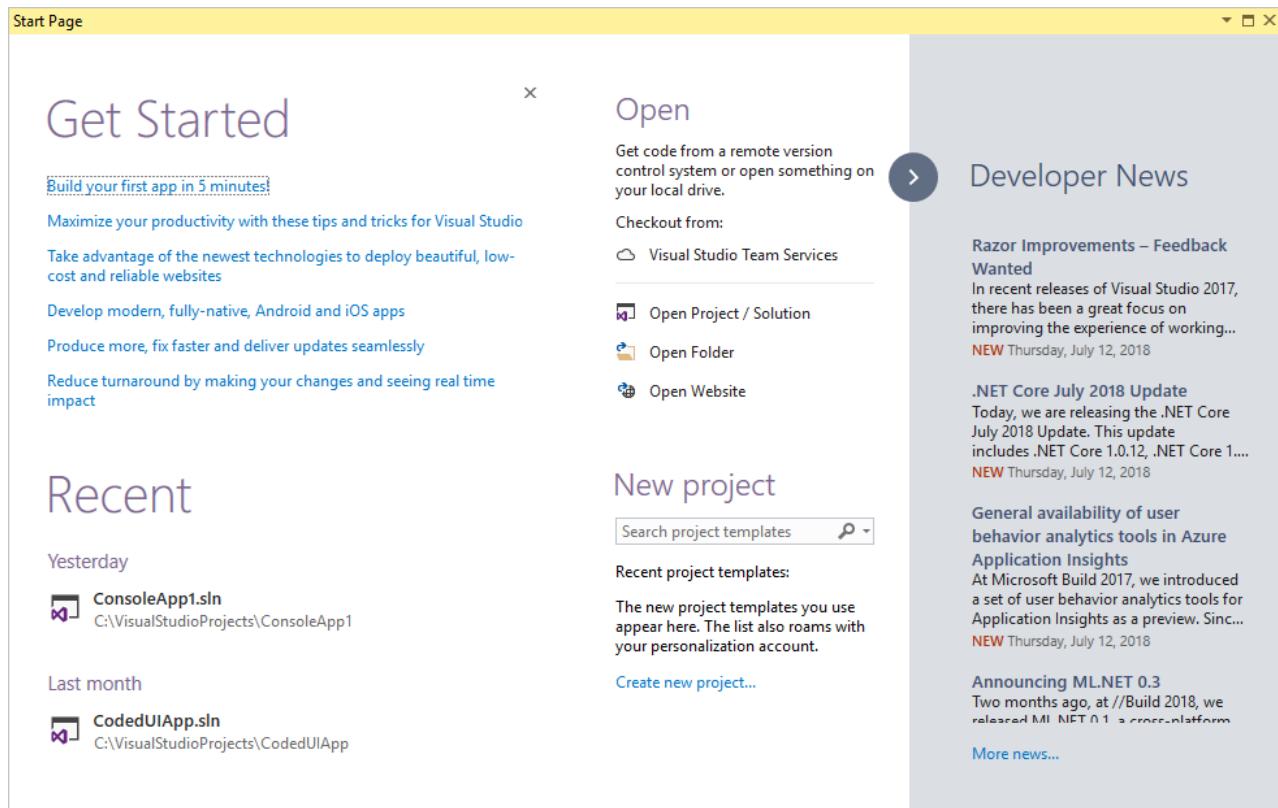
In this 5-10 minute introduction to the Visual Studio integrated development environment (IDE), we'll take a tour of some of the windows, menus, and other UI features.

If you haven't already installed Visual Studio, go to the [Visual Studio downloads](#) page to install it for free.

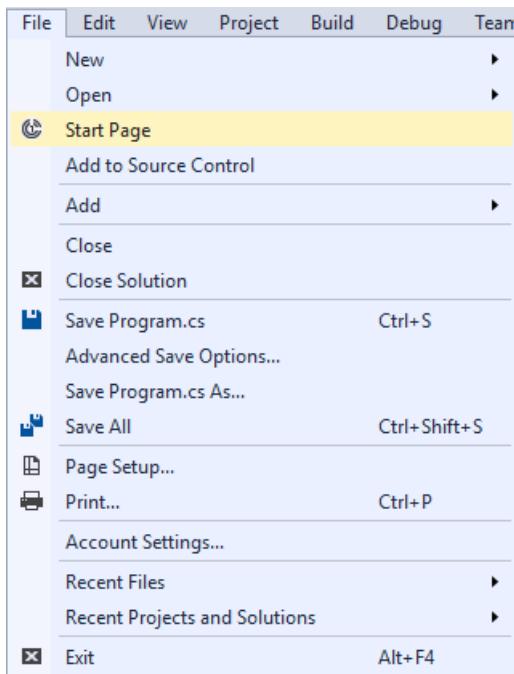
If you haven't already installed Visual Studio, go to the [Visual Studio downloads](#) page to install it for free.

Start Page

The first thing you'll see after you open Visual Studio is most likely the **Start Page**. The **Start Page** is designed as a "hub" to help you find the commands and project files you need faster. The **Recent** section displays projects and folders you've worked on recently. Under **New project**, you can click a link to bring up the **New Project** dialog box, or under **Open**, you can open an existing code project or folder. On the right is a feed of the latest developer news.

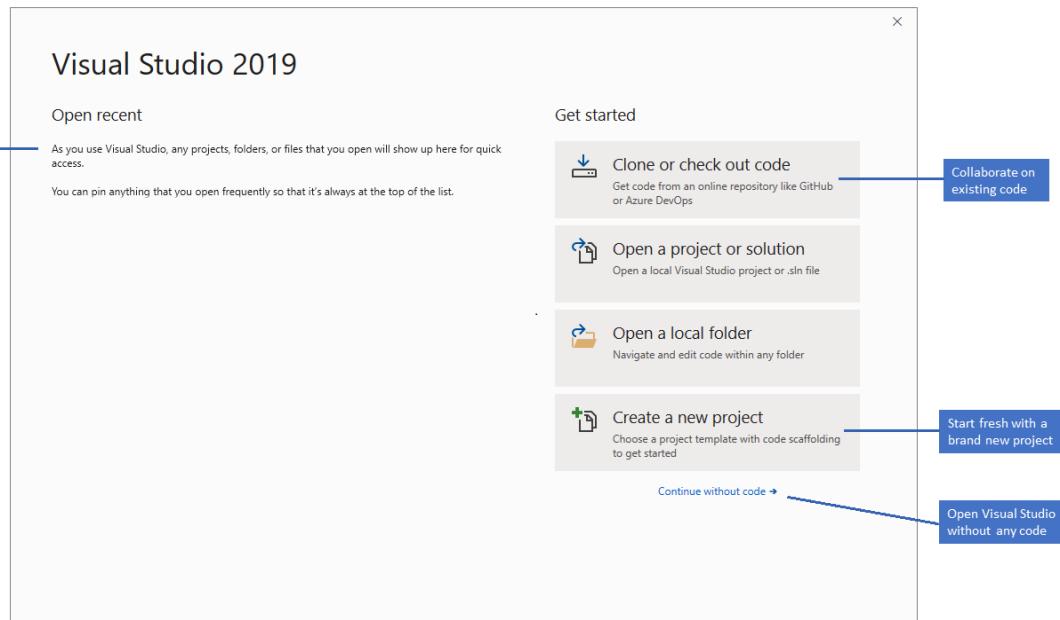


If you close the **Start Page** and want to see it again, you can reopen it from the **File** menu.



Start window

The first thing you'll see after you open Visual Studio is the start window. The start window is designed to help you "get to code" faster. It has options to clone or check out code, open an existing project or solution, create a new project, or simply open a folder that contains some code files.



If this is the first time you're using Visual Studio, your recent projects list will be empty.

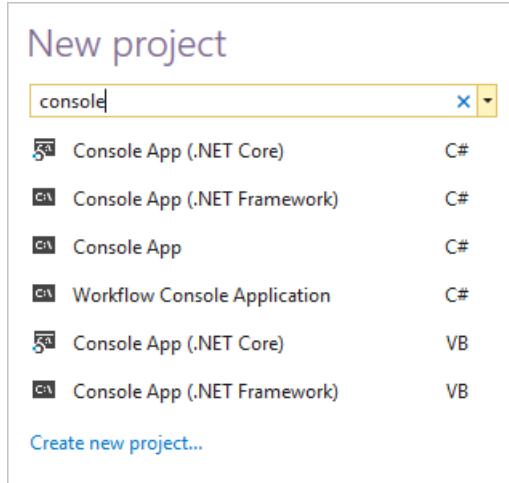
If you work with non-MSBuild based codebases, you'll use the **Open a local folder** option to open your code in Visual Studio. For more information, see [Develop code in Visual Studio without projects or solutions](#). Otherwise, you can create a new project or clone a project from a source provider such as GitHub or Azure DevOps.

The **Continue without code** option simply opens the Visual Studio development environment without any specific project or code loaded. You might choose this option to join a [Live Share](#) session or attach to a process for debugging. You can also press **Esc** to close the start window and open the IDE.

Create a project

To continue exploring Visual Studio's features, let's create a new project.

1. On the **Start Page**, in the search box under **New project**, type in **console** to filter the list of project types to those that contain "console" in their name.



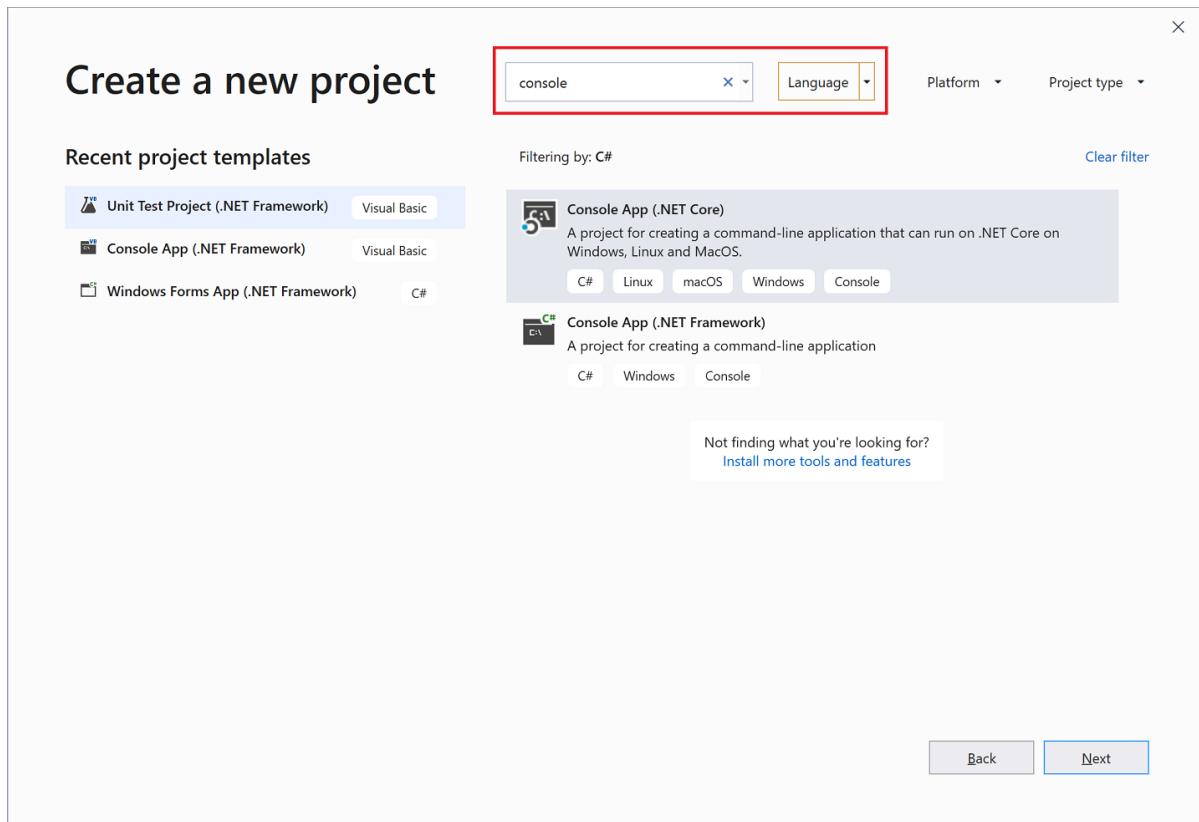
Visual Studio provides various kinds of project templates that help you get started coding quickly. Choose a **C# Console App (.NET Core)** project template. (Alternatively, if you're a Visual Basic, C++, Javascript, or other language developer, feel free to create a project in one of those languages. The UI we'll be looking at is similar for all programming languages.)

2. In the **New Project** dialog box that appears, accept the default project name and choose **OK**.

1. On the start window, choose **Create a new project**.

A dialog box opens that says **Create a new project**. Here, you can search, filter, and pick a project template. It also shows a list of your recently used project templates.

2. In the search box at the top, type in **console** to filter the list of project types to those that contain "console" in their name. Further refine the search results by picking **C#** (or another language of your choice) from the **Language** picker.



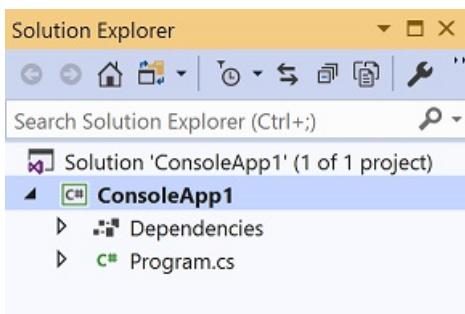
3. If you selected C#, Visual Basic, or F# as your language, select the **Console App (.NET Core)** template, and then choose **Next**. (If you selected a different language, just pick any template. The UI we'll be looking at is similar for all programming languages.)
4. On the **Configure your new project** page, accept the default project name and location, and then choose **Create**.

The project is created and a file named *Program.cs* opens in the **Editor** window. The **Editor** shows the contents of files and is where you'll do most of your coding work in Visual Studio.

```
ConsoleApp1 - Program.cs
Program.cs  X
C# ConsoleApp1  ConsoleApp1.Program  Main(string[] args)
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6
7  namespace ConsoleApp1
8  {
9      class Program
10     {
11         static void Main(string[] args)
12         {
13         }
14     }
15 }
16
```

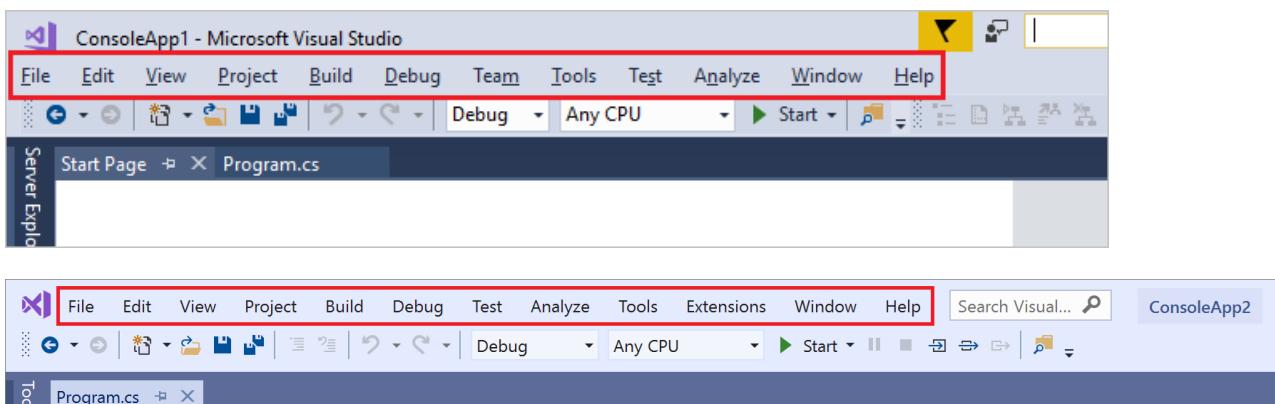
Solution Explorer

Solution Explorer, which is typically on the right-hand side of Visual Studio, shows you a graphical representation of the hierarchy of files and folders in your project, solution, or code folder. You can browse the hierarchy and navigate to a file in **Solution Explorer**.



Menus

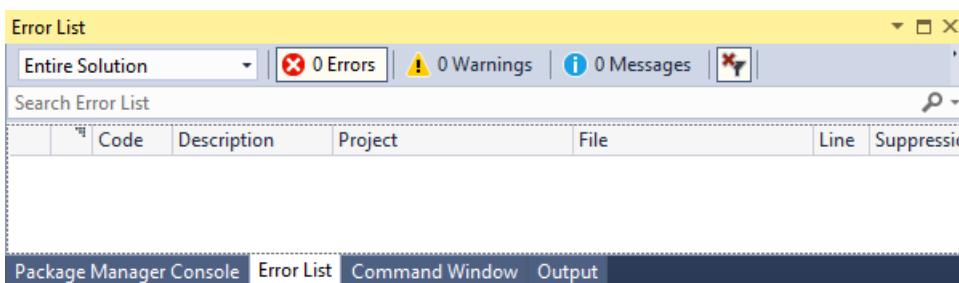
The menu bar along the top of Visual Studio groups commands into categories. For example, the **Project** menu contains commands related to the project you're working in. On the **Tools** menu, you can customize how Visual Studio behaves by selecting **Options**, or add features to your installation by selecting **Get Tools and Features**.



Error List

Open the **Error List** window by choosing the **View** menu, and then **Error List**.

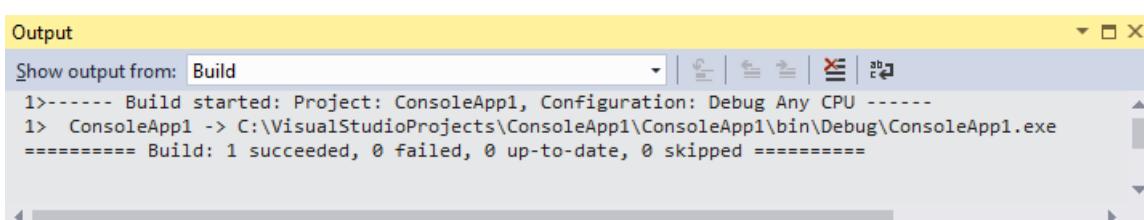
The **Error List** shows you errors, warning, and messages regarding the current state of your code. If there are any errors (such as a missing brace or semicolon) in your file, or anywhere in your project, they're listed here.



Output window

The **Output** window shows you output messages from building your project and from your source control provider.

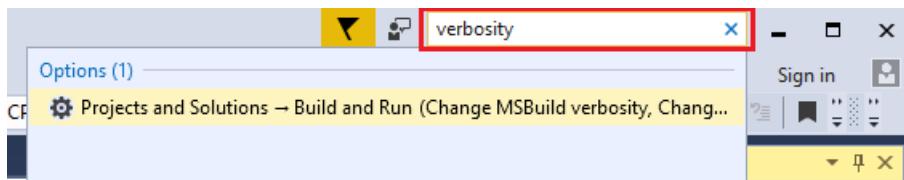
Let's build the project to see some build output. From the **Build** menu, choose **Build Solution**. The **Output** window automatically obtains focus and display a successful build message.



Search box

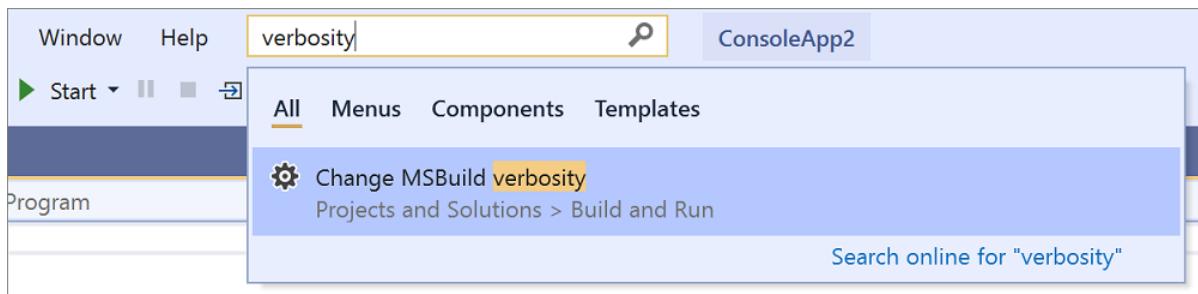
The search box is a quick and easy way to navigate to pretty much anything in Visual Studio. You can enter some text related to what you want to do, and it'll show you a list of options that pertain to the text. For example, imagine you want to increase the build output's verbosity to display additional details about what exactly build is doing. Here's how you might do that:

1. Locate the **Quick Launch** search box in the upper right of the IDE. (Alternatively, press **Ctrl+Q** to access it.)
2. Type **verbosity** into the search box. From the displayed results, choose **Projects and Solutions --> Build and Run** under the **Options** category.



The **Options** dialog box opens to the **Build and Run** options page.

1. Press **Ctrl+Q** to activate the search box in the upper part of the IDE.
2. Type **verbosity** into the search box. From the displayed results, choose **Change MSBuild verbosity**.



The **Options** dialog box opens to the **Build and Run** options page.

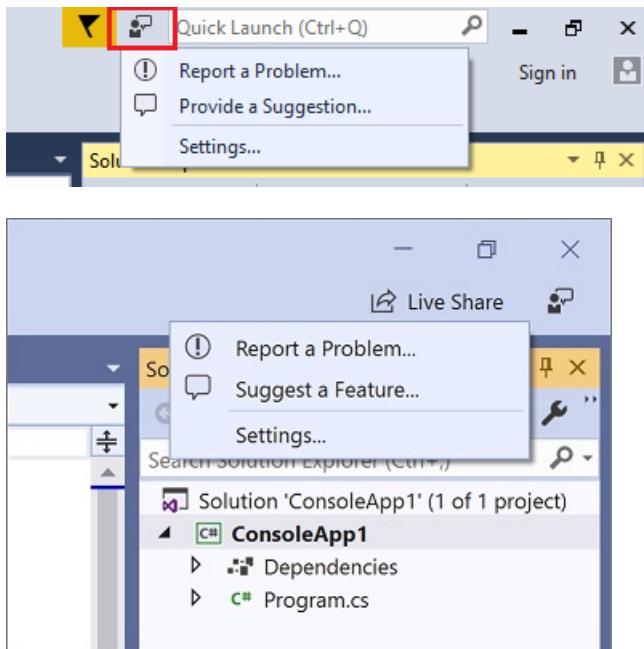
3. Under **MSBuild project build output verbosity**, choose **Normal**, and then click **OK**.
4. Build the project again by right-clicking on the **ConsoleApp1** project in **Solution Explorer** and choosing **Rebuild** from the context menu.

This time the **Output** window shows more verbose logging from the build process, including which files were copied where.

```
Output
Show output from: Build
1>----- Build started: Project: ConsoleApp1, Configuration: Debug Any CPU -----
1>Build started 7/13/2018 1:50:24 PM.
1>GenerateBindingRedirects:
1> No suggested binding redirects from ResolveAssemblyReferences.
1>GenerateTargetFrameworkMonikerAttribute:
1>Skipping target "GenerateTargetFrameworkMonikerAttribute" because all output files are up-to-date
1>CoreCompile:
1> C:\Program Files (x86)\Microsoft Visual Studio\2017\Enterprise\MSBuild\15.0\Bin\Roslyn\csc.exe /
1> Using shared compilation with compiler from directory: C:\Program Files (x86)\Microsoft Visual S
1>_CopyAppConfigFile:
1> Copying file from "App.config" to "bin\Debug\ConsoleApp1.exe.config".
1>CopyFilesToOutputDirectory:
1> Copying file from "obj\Debug\ConsoleApp1.exe" to "bin\Debug\ConsoleApp1.exe".
1> ConsoleApp1 -> C:\VisualStudioProjects\ConsoleApp1\ConsoleApp1\bin\Debug\ConsoleApp1.exe
1> Copying file from "obj\Debug\ConsoleApp1.pdb" to "bin\Debug\ConsoleApp1.pdb".
1>
1>Build succeeded.
1> 0 Warning(s)
1> 0 Error(s)
1>
1>Time Elapsed 00:00:00.06
===== Build: 1 succeeded, 0 failed, 0 up-to-date, 0 skipped =====
```

Send Feedback menu

Should you encounter any problems while you're using Visual Studio, or if you have suggestions for how to improve the product, you can use the **Send Feedback** menu near the top of the Visual Studio window.



Next steps

We've looked at just a few of the features of Visual Studio to get acquainted with the user interface. To explore further:

[Learn about the code editor](#)

[Learn about projects and solutions](#)

See also

- [Overview of the Visual Studio IDE](#)
- [More features of Visual Studio](#)
- [Change theme and font colors](#)

Quickstart: Create your first Python web app using Visual Studio

8/30/2019 • 7 minutes to read • [Edit Online](#)

In this 5-10 minute introduction to Visual Studio as a Python IDE, you create a simple Python web application based on the Flask framework. You create the project through discrete steps that help you learn about Visual Studio's basic features.

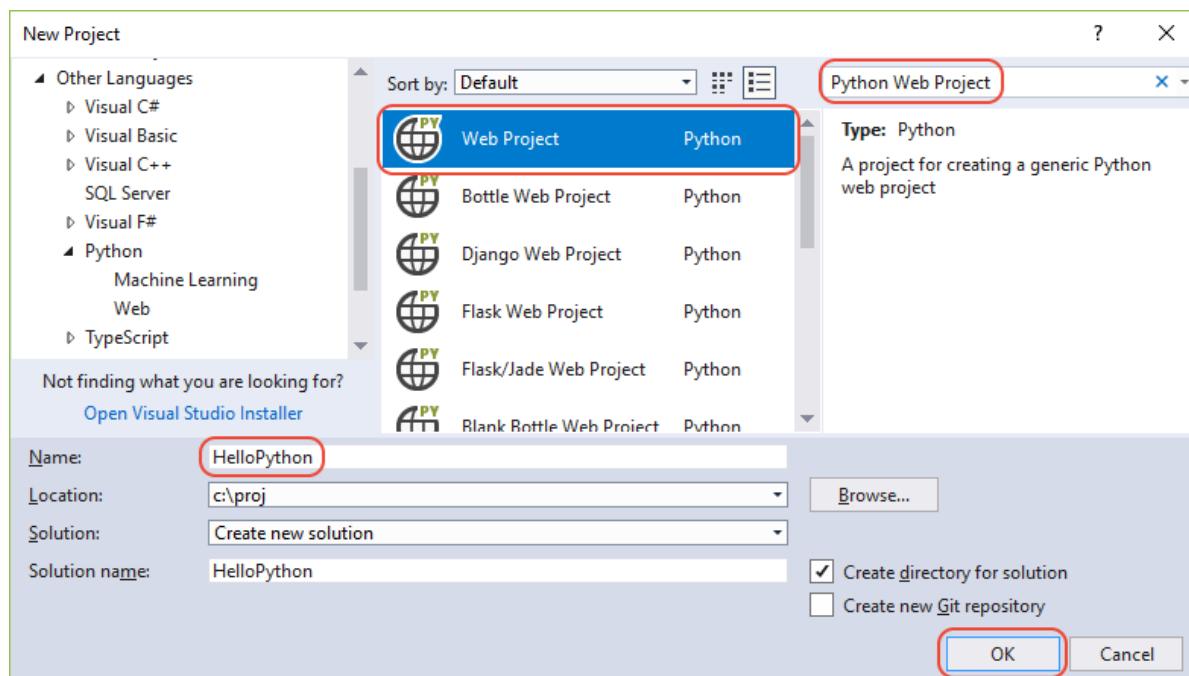
If you haven't already installed Visual Studio, go to the [Visual Studio downloads](#) page to install it for free. In the installer, make sure to select the **Python development** workload.

If you haven't already installed Visual Studio, go to the [Visual Studio downloads](#) page to install it for free. In the installer, make sure to select the **Python development** workload.

Create the project

The following steps create an empty project that serves as a container for the application:

1. Open Visual Studio 2017.
2. From the top menu bar, choose **File > New > Project**.
3. In the **New Project** dialog box, enter "Python Web Project" in the search field on the upper right, choose **Web project** in the middle list, give the project a name like "HelloPython", then choose **OK**.

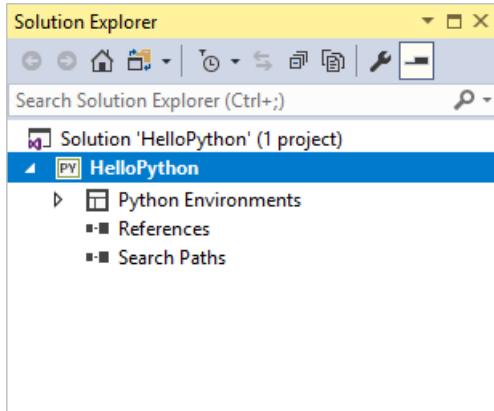


If you don't see the Python project templates, run the **Visual Studio Installer**, select **More > Modify**, select the **Python development** workload, then choose **Modify**.

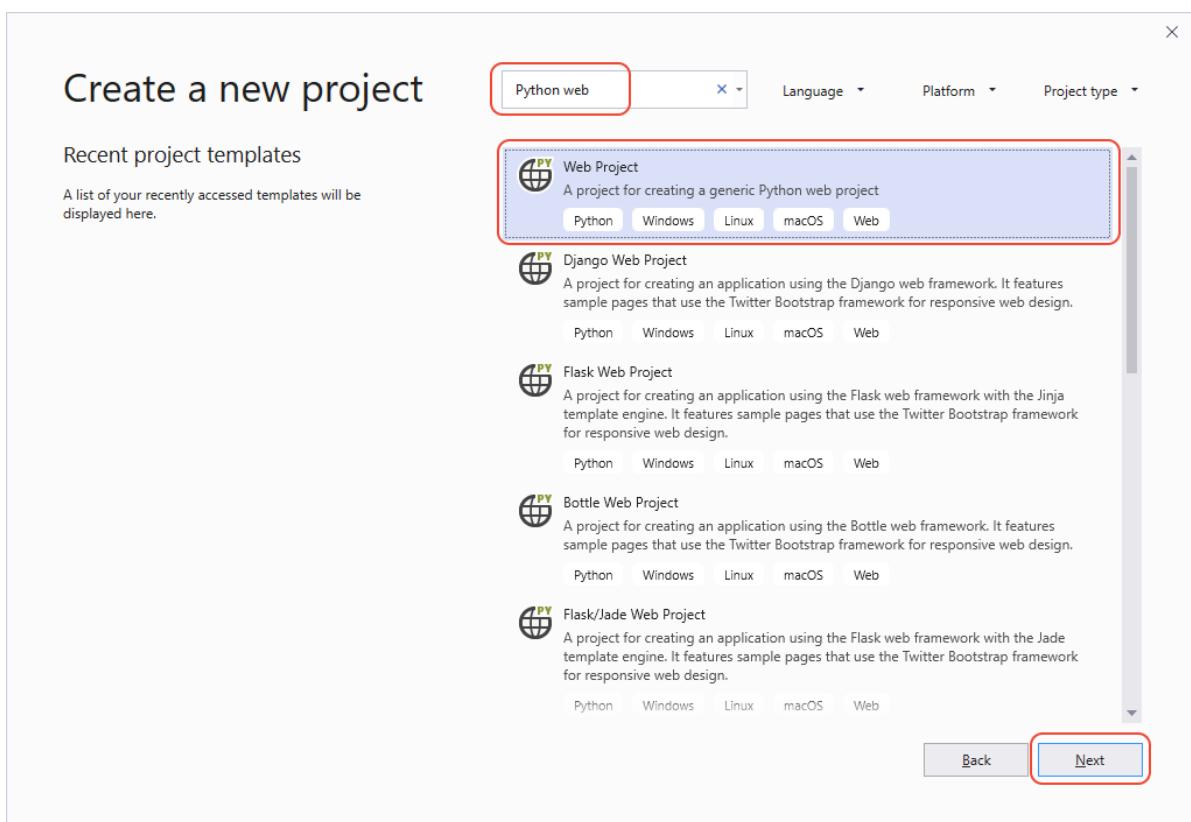


4. The new project opens in **Solution Explorer** in the right pane. The project is empty at this point because it

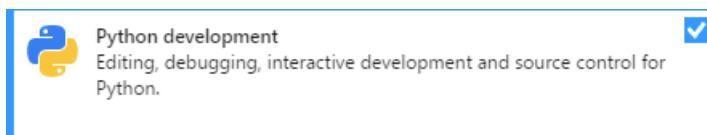
contains no other files.



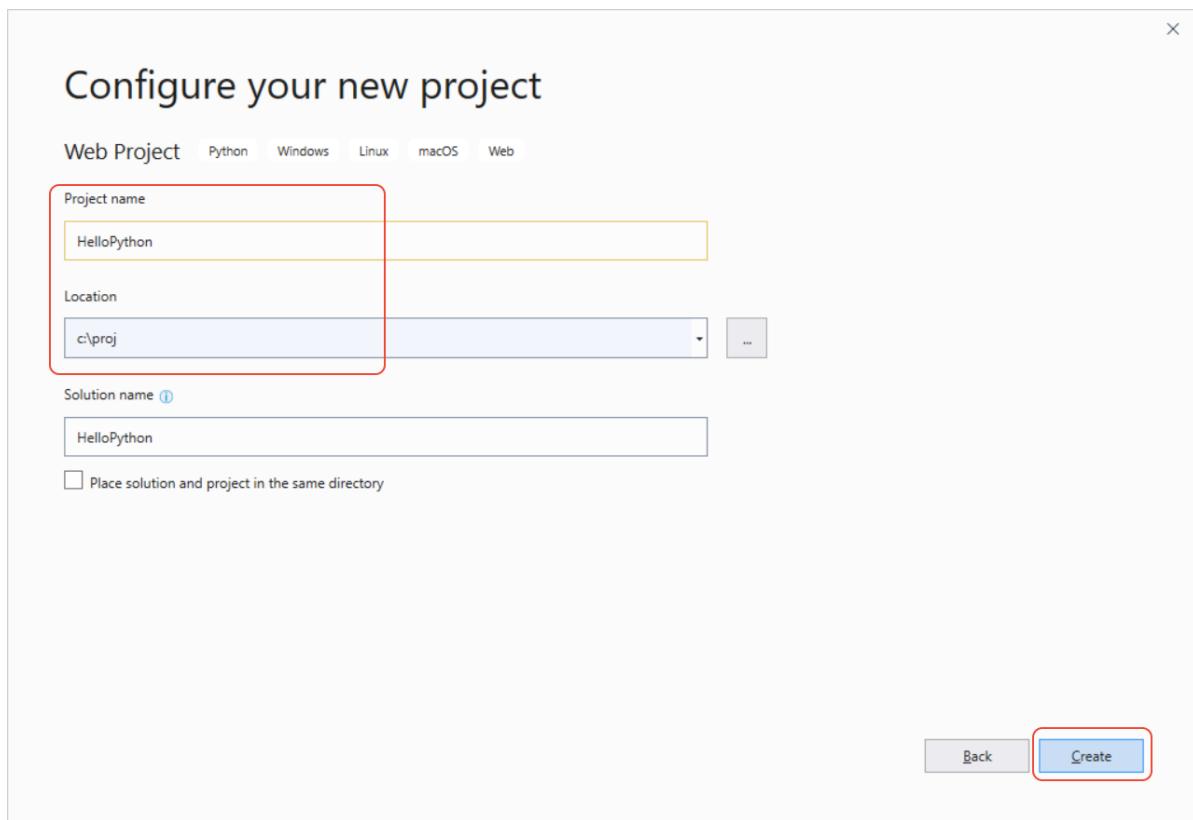
1. Open Visual Studio 2019.
2. On the start screen, select **Create a new project**.
3. In the **Create a new project** dialog box, enter "Python web" in the search field at the top, choose **Web Project** in the middle list, then select **Next**:



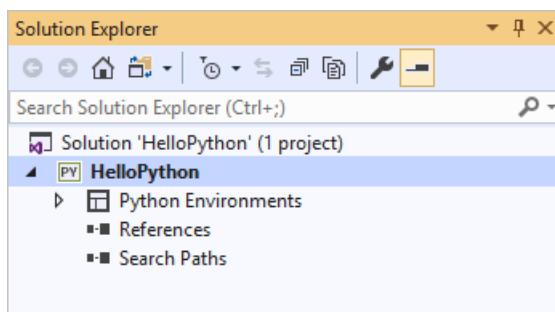
If you don't see the Python project templates, run the **Visual Studio Installer**, select **More > Modify**, select the **Python development** workload, then choose **Modify**.



4. In the **Configure your new project** dialog that follows, enter "HelloPython" for **Project name**, specify a location, and select **Create**. (The **Solution name** is automatically set to match the **Project name**.)



5. The new project opens in **Solution Explorer** in the right pane. The project is empty at this point because it contains no other files.



Question: What's the advantage of creating a project in Visual Studio for a Python application?

Answer: Python applications are typically defined using only folders and files, but this simple structure can become burdensome as applications become larger and perhaps involve auto-generated files, JavaScript for web applications, and so on. A Visual Studio project helps manage this complexity. The project (a `.pyproj` file) identifies all the source and content files associated with your project, contains build information for each file, maintains the information to integrate with source-control systems, and helps you organize your application into logical components.

Question: What is the "solution" shown in Solution Explorer?

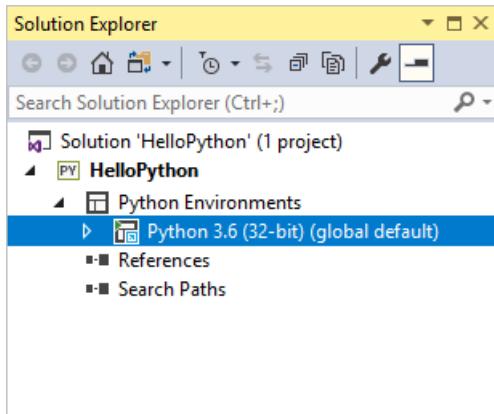
Answer: A Visual Studio solution is a container that helps you manage for one or more related projects as a group, and stores configuration settings that aren't specific to a project. Projects in a solution can also reference one another, such that running one project (a Python app) automatically builds a second project (such as a C++ extension used in the Python app).

Install the Flask library

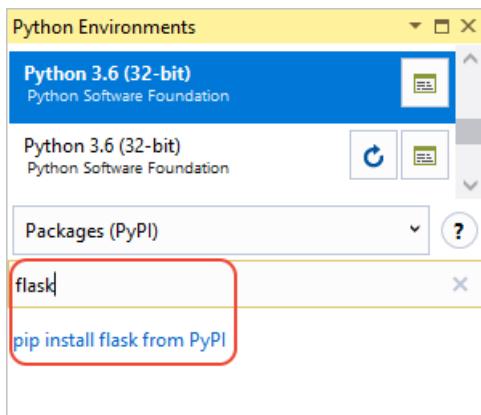
Web apps in Python almost always use one of the many available Python libraries to handle low-level details like routing web requests and shaping responses. For this purpose, Visual Studio provides a variety of templates for web apps, one of which you use later in this Quickstart.

Here, you use the following steps to install the Flask library into the default "global environment" that Visual Studio uses for this project.

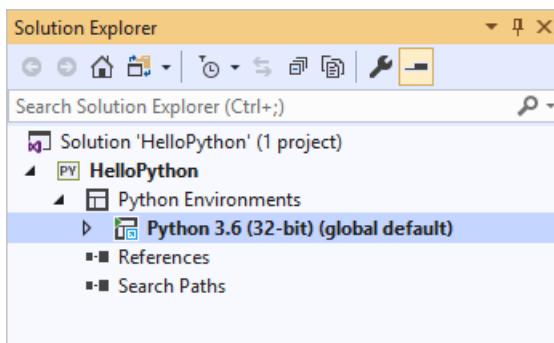
1. Expand the **Python Environments** node in the project to see the default environment for the project.



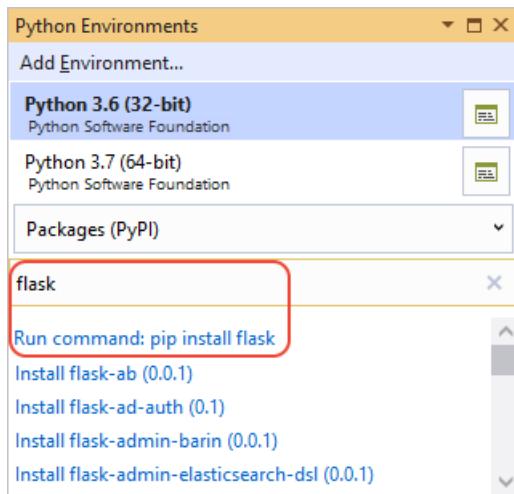
2. Right-click the environment and select **Install Python Package**. This command opens the **Python Environments** window on the **Packages** tab.
3. Enter "flask" in the search field and select **pip install flask from PyPI**. Accept any prompts for administrator privileges and observe the **Output** window in Visual Studio for progress. (A prompt for elevation happens when the packages folder for the global environment is located within a protected area like C:\Program Files.)



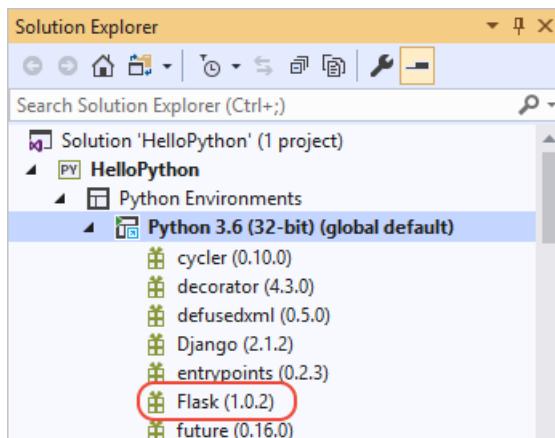
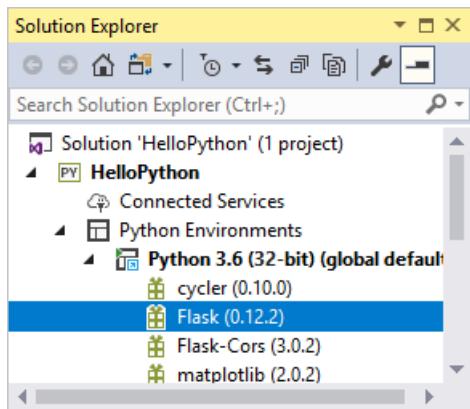
1. Expand the **Python Environments** node in the project to see the default environment for the project.



2. Right-click the environment and select **Manage Python Packages...**. This command opens the **Python Environments** window on the **Packages (PyPI)** tab.
3. Enter "flask" in the search field. If **Flask** appears below the search box, you can skip this step. Otherwise select **Run command: pip install flask**. Accept any prompts for administrator privileges and observe the **Output** window in Visual Studio for progress. (A prompt for elevation happens when the packages folder for the global environment is located within a protected area like C:\Program Files.)



- Once installed, the library appears in the environment in **Solution Explorer**, which means that you can make use of it in Python code.



NOTE

Instead of installing libraries in the global environment, developers typically create a "virtual environment" in which to install libraries for a specific project. Visual Studio templates typically offer this option, as discussed in [Quickstart - Create a Python project using a template](#).

Question: Where do I learn more about other available Python packages?

Answer: Visit the [Python Package Index](#).

Add a code file

You're now ready to add a bit of Python code to implement a minimal web app.

1. Right-click the project in **Solution Explorer** and select **Add > New Item**.
2. In the dialog that appears, select **Empty Python File**, name it *app.py*, and select **Add**. Visual Studio automatically opens the file in an editor window.
3. Copy the following code and paste it into *app.py*:

```
from flask import Flask

# Create an instance of the Flask class that is the WSGI application.
# The first argument is the name of the application module or package,
# typically __name__ when using a single module.
app = Flask(__name__)

# Flask route decorators map / and /hello to the hello function.
# To add other resources, create functions that generate the page contents
# and add decorators to define the appropriate resource locators for them.

@app.route('/')
@app.route('/hello')
def hello():
    # Render the page
    return "Hello Python!"

if __name__ == '__main__':
    # Run the app server on localhost:4449
    app.run('localhost', 4449)
```

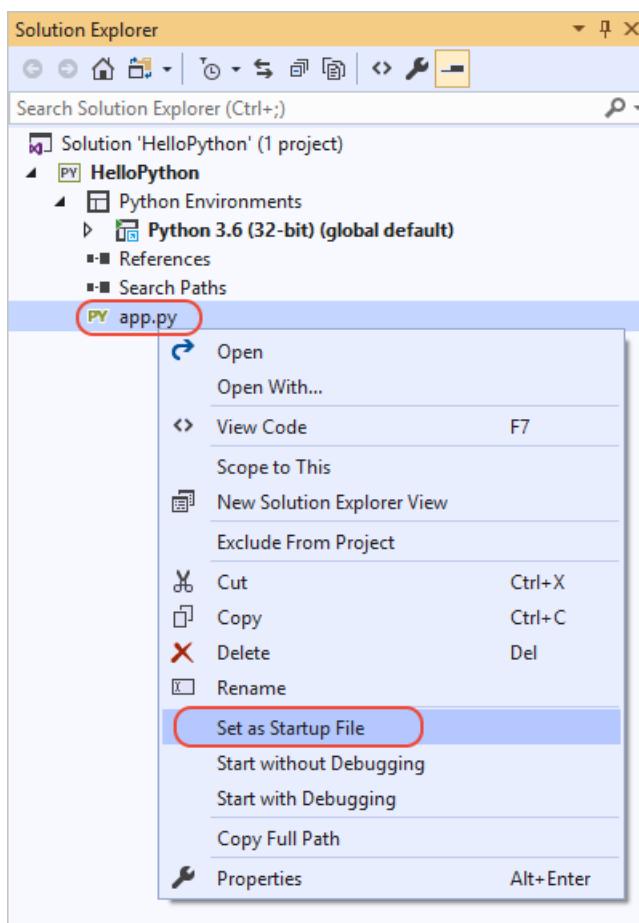
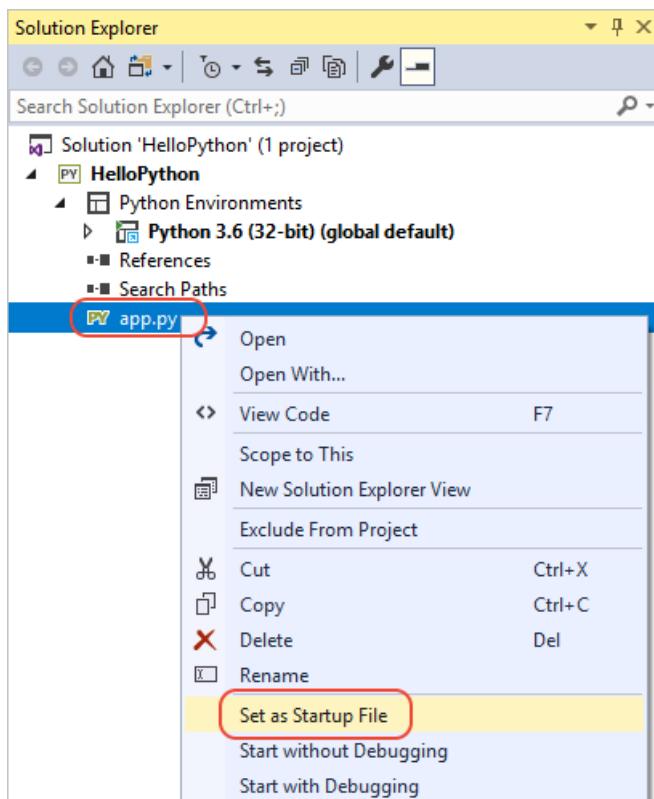
4. You may have noticed that the **Add > New Item** dialog box displays many other types of files you can add to a Python project, including a Python class, a Python package, a Python unit test, *web.config* files, and more. In general, these item templates, as they're called, are a great way to quickly create files with useful boilerplate code.

Question: Where can I learn more about Flask?

Answer: Refer to the Flask documentation, starting with the [Flask Quickstart](#).

Run the application

1. Right-click *app.py* in **Solution Explorer** and select **Set as startup file**. This command identifies the code file to launch in Python when running the app.



2. Right-click the project in **Solution Explorer** and select **Properties**. Then select the **Debug** tab and set the **Port Number** property to `4449`. This step ensures that Visual Studio launches a browser with `localhost:4449` to match the `app.run` arguments in the code.
3. Select **Debug > Start Without Debugging (Ctrl+F5)**, which saves changes to files and runs the app.
4. A command window appears with the message "`* Running in https://localhost:4449/`", and a browser window should open to `localhost:4449` where you see the message, "Hello, Python!" The GET request also

appears in the command window with a status of 200.

If a browser does not open automatically, start the browser of your choice and navigate to `localhost:4449`.

If you see only the Python interactive shell in the command window, or if that window flashes on the screen briefly, ensure that you set `app.py` as the startup file in step 1 above.

5. Navigate to `localhost:4449/hello` to test that the decorator for the `/hello` resource also works. Again, the GET request appears in the command window with a status of 200. Feel free to try some other URL as well to see that they show 404 status codes in the command window.
6. Close the command window to stop the app, then close the browser window.

Question: What's the difference between the Start Without Debugging command and Start Debugging?

Answer: You use **Start Debugging** to run the app in the context of the [Visual Studio debugger](#), allowing you to set breakpoints, examine variables, and step through your code line by line. Apps may run slower in the debugger because of the various hooks that make debugging possible. **Start Without Debugging**, in contrast, runs the app directly as if you ran it from the command line, with no debugging context, and also automatically launches a browser and navigates to the URL specified in the project properties' **Debug** tab.

Next steps

Congratulations on running your first Python app from Visual Studio, in which you've learned a little about using Visual Studio as a Python IDE!

[Deploy the app to Azure App Service](#)

Because the steps you followed in this Quickstart are fairly generic, you've probably guessed that they can and should be automated. Such automation is the role of Visual Studio project templates. Go through [Quickstart - Create a Python project using a template](#) for a demonstration that creates a web app similar to the one you created in this article, but with fewer steps.

To continue with a fuller tutorial on Python in Visual Studio, including using the interactive window, debugging, data visualization, and working with Git, go through [Tutorial: Get started with Python in Visual Studio](#).

To explore more that Visual Studio has to offer, select the links below.

- Learn about [Python web app templates in Visual Studio](#).
- Learn about [Python debugging](#)
- Learn more about the [Visual Studio IDE](#) in general.

Quickstart: Use Visual Studio to create your first Node.js app

8/30/2019 • 3 minutes to read • [Edit Online](#)

In this 5-10 minute introduction to the Visual Studio integrated development environment (IDE), you'll create a simple Node.js web application.

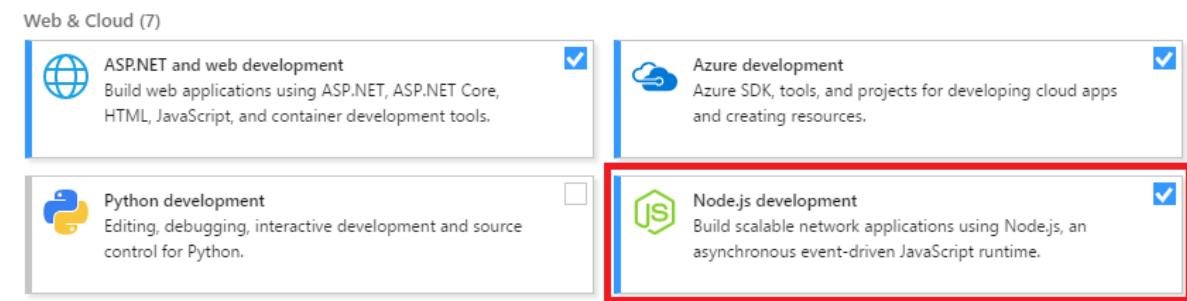
Prerequisites

- You must have Visual Studio installed and the Node.js development workload.

If you haven't already installed Visual Studio 2019, go to the [Visual Studio downloads](#) page to install it for free.

If you haven't already installed Visual Studio 2017, go to the [Visual Studio downloads](#) page to install it for free.

If you need to install the workload but already have Visual Studio, go to **Tools > Get Tools and Features...**, which opens the Visual Studio Installer. Choose the **Node.js development** workload, then choose **Modify**.



- You must have the Node.js runtime installed.

If you don't have it installed, install the LTS version from the [Node.js](#) website. In general, Visual Studio automatically detects the installed Node.js runtime. If it does not detect an installed runtime, you can configure your project to reference the installed runtime in the properties page (after you create a project, right-click the project node and choose **Properties**).

Create a project

First, you'll create an Node.js web application project.

1. If you don't have the Node.js runtime already installed, install the LTS version from the [Node.js](#) website.

In general, Visual Studio automatically detects the installed Node.js runtime. If it does not detect an installed runtime, you can configure your project to reference the installed runtime in the properties page (after you create a project, right-click the project node and choose **Properties**).

2. Open Visual Studio.

3. Create a new project.

Press **Esc** to close the start window. Type **Ctrl + Q** to open the search box, type **Node.js**, then choose **Create new Blank Node.js Web application project** (JavaScript). In the dialog box that appears, choose **Create**.

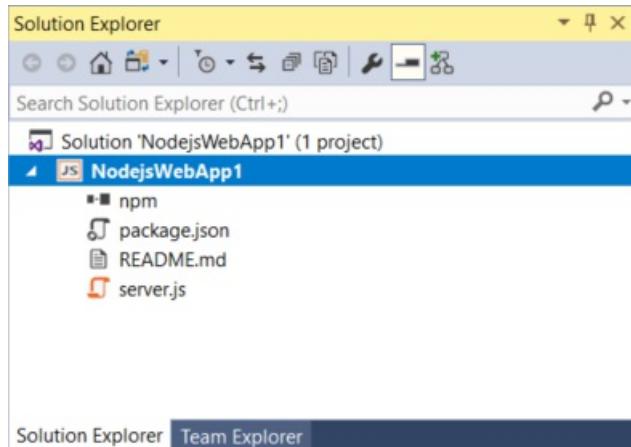
From the top menu bar, choose **File > New > Project**. In the left pane of the **New Project** dialog box, expand **JavaScript**, then choose **Node.js**. In the middle pane, choose **Blank Node.js Web application**, then choose **OK**.

If you don't see the **Blank Node.js Web application** project template, you must add the **Node.js development** workload. For detailed instructions, see the [Prerequisites](#).

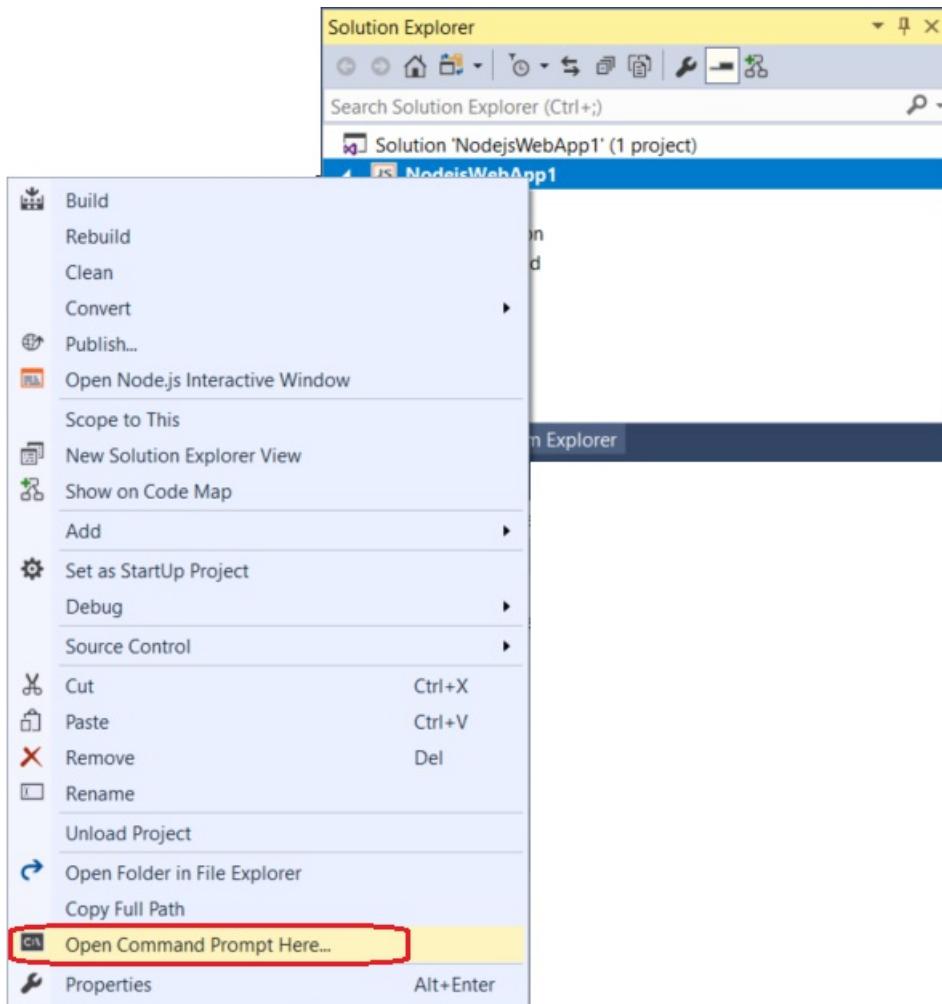
Visual Studio creates and the new solution and opens the project. *server.js* opens in the editor in the left pane.

Explore the IDE

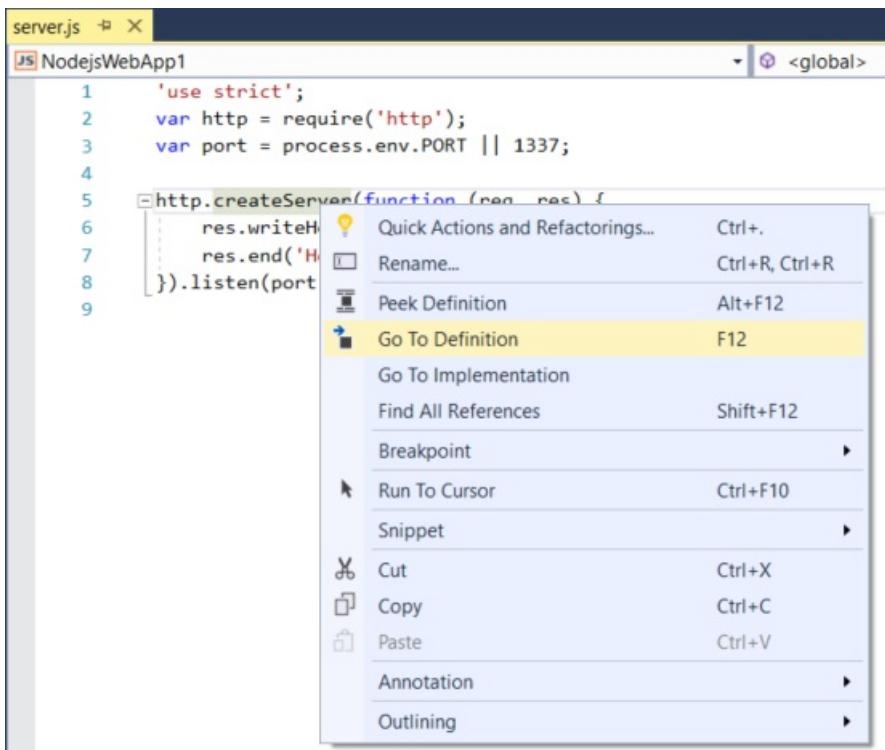
1. Take a look at **Solution Explorer** in the right pane.



- Highlighted in bold is your project, using the name you gave in the **New Project** dialog box. On disk, this project is represented by a *.njsproj* file in your project folder.
 - At the top level is a solution, which by default has the same name as your project. A solution, represented by a *.sln* file on disk, is a container for one or more related projects.
 - The npm node shows any installed npm packages. You can right-click the npm node to search for and install npm packages using a dialog box.
2. If you want to install npm packages or Node.js commands from a command prompt, right-click the project node and choose **Open Command Prompt Here**.



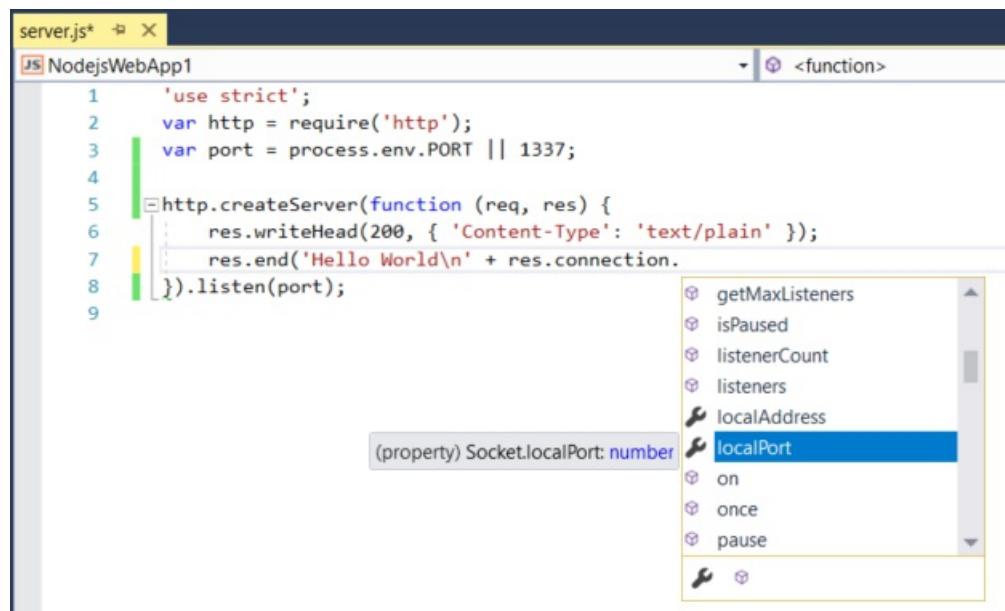
- In the `server.js` file in the editor (left pane), choose `http.createServer` and then press **F12** or choose **Go To Definition** from the context (right-click) menu. This command takes you to the definition of the `createServer` function in `index.d.ts`.



- Got back to `server.js`, then put your cursor at the end of the string in this line of code, `res.end('Hello World\n');`, and modify it so that it looks like this:

```
res.end('Hello World\n' + res.connection.
```

Where you type `connection.`, IntelliSense provides options to auto-complete the code entry.



- Choose **localPort**, and then type `);` to complete the statement so that it looks like this:

```
res.end('Hello World\n' + res.connection.localPort);
```

Run the application

- Press **Ctrl+F5** (or **Debug > Start Without Debugging**) to run the application. The app opens in a browser.
- In the browser window, you will see "Hello World" plus the local port number.
- Close the web browser.

Congratulations on completing this Quickstart in which you got started with the Visual Studio IDE and Node.js. If you'd like to delve deeper into its capabilities, continue with a tutorial in the **Tutorials** section of the table of contents.

Next steps

[Deploy the app to Linux App Service](#)

- [Tutorial for Node.js and Express](#)
- [Tutorial for Node.js and React](#)

Quickstart: Use Visual Studio to create your first ASP.NET Core web service in F#

8/30/2019 • 2 minutes to read • [Edit Online](#)

In this 5-10 minute introduction to F# in Visual Studio, you'll create an F# ASP.NET Core web application.

If you haven't already installed Visual Studio, go to the [Visual Studio downloads](#) page to install it for free.

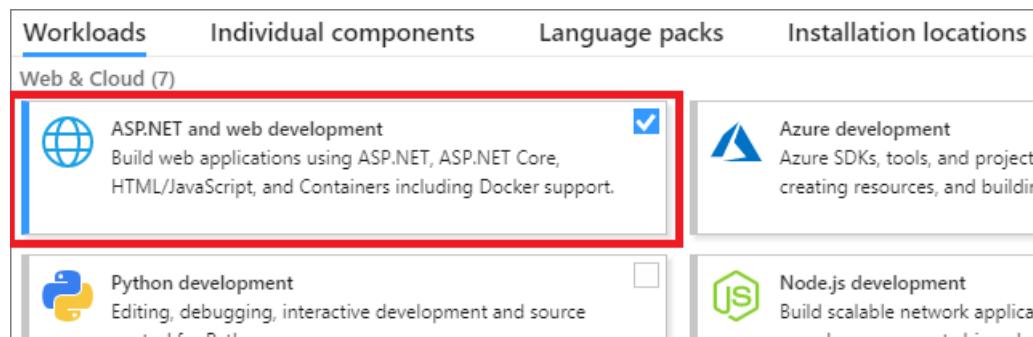
If you haven't already installed Visual Studio, go to the [Visual Studio downloads](#) page to install it for free.

Create a project

First, you'll create an ASP.NET Core Web API project. The project type comes with template files that constitute a functional web service, before you've even added anything!

1. Open Visual Studio.
2. From the top menu bar, choose **File > New > Project**.
3. In the **New Project** dialog box, in the left pane, expand **Visual F#**, then choose **Web**. In the middle pane, choose **ASP.NET Core Web Application**, then choose **OK**.

If you don't see the **.NET Core** project template category, choose the [Open Visual Studio Installer](#) link in the left pane. The Visual Studio Installer launches. Choose the **ASP.NET and web development** workload, then choose **Modify**.

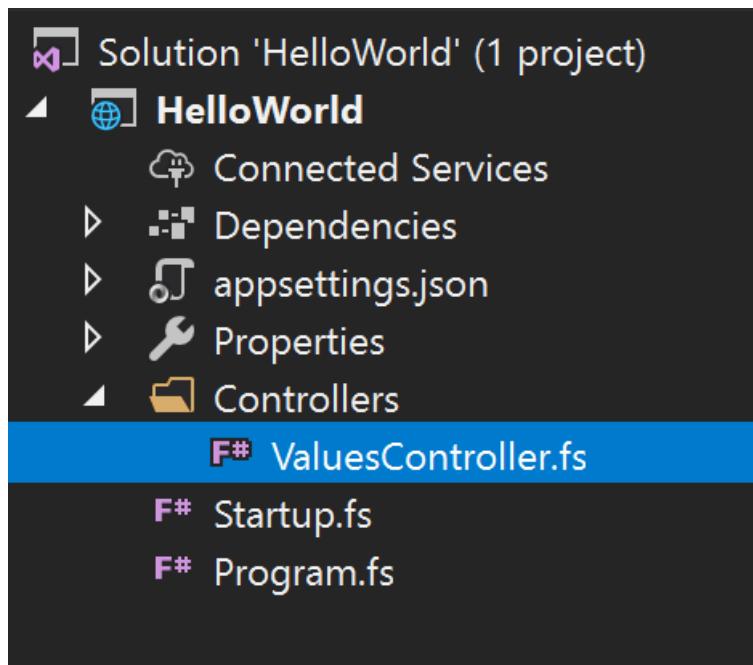


4. In the **New ASP.NET Core Web Application** dialog box, select **ASP.NET Core 2.1** from the top drop-down menu. (If you don't see **ASP.NET Core 2.1** in the list, install it by following the [Download](#) link that should appear in a yellow bar near the top of the dialog box.) Choose **OK**.

1. Open Visual Studio.
2. On the start window, choose **Create a new project**.
3. On the **Create a new project** page, type **f# web** into the search box, and then choose the **ASP.NET Core Web Application** project template. Choose **Next**.
4. On the **Configure your new project** page, enter a name, and then choose **Create**.
5. On the **Create a new ASP.NET Core Web Application** page, select **ASP.NET Core 2.1** from the top drop-down menu, and then choose **Create**.

Explore the IDE

1. In the **Solution Explorer** toolbar, expand the **Controllers** folder, then choose **ValuesController.fs** to open it in the editor.



2. Next, modify the `Get()` member to be the following:

```
[<HttpGet>]
member this.Get() =
    let values = [| "Hello"; "World"; "First F#/ASP.NET Core web API!" |]
    ActionResult<string[]>(values)
```

The code is straightforward. An F# array of values is bound to the `values` name, and then passed to the ASP.NET Core MVC framework as an `ActionResult`. ASP.NET Core takes care of the rest for you.

It should look like this in the editor:

```
[<HttpGet>]
member this.Get() =
    let values = [| "Hello"; "World"; "First F#/ASP.NET Core web API!" |]
    ActionResult<string[]>(values)
```

Run the application

1. Press **Ctrl+F5** to run the application and open it in a web browser.
2. The page should navigate to the `/api/values` route, but if it does not, enter `https://localhost:44396/api/values` into your browser.

The web browser will now display JSON matching what you typed earlier.

Next steps

Congratulations on completing this Quickstart! We hope you learned a little bit about F#, ASP.NET Core, and the Visual Studio IDE. To see the app running on a public server, select the following button.

[Deploy the app to Azure App Service](#)

To learn more about F#, check out the official [F# Guide](#).

Quickstart: Use Visual Studio to create your first ASP.NET Core web app

11/7/2019 • 4 minutes to read • [Edit Online](#)

In this 5-10 minute introduction to how to use Visual Studio, you'll create a simple "Hello World" web app by using an ASP.NET project template and the C# programming language.

Before you begin

Install Visual Studio

If you haven't already installed Visual Studio, go to the [Visual Studio downloads](#) page to install it for free.

If you haven't already installed Visual Studio, go to the [Visual Studio downloads](#) page to install it for free.

Choose your theme (optional)

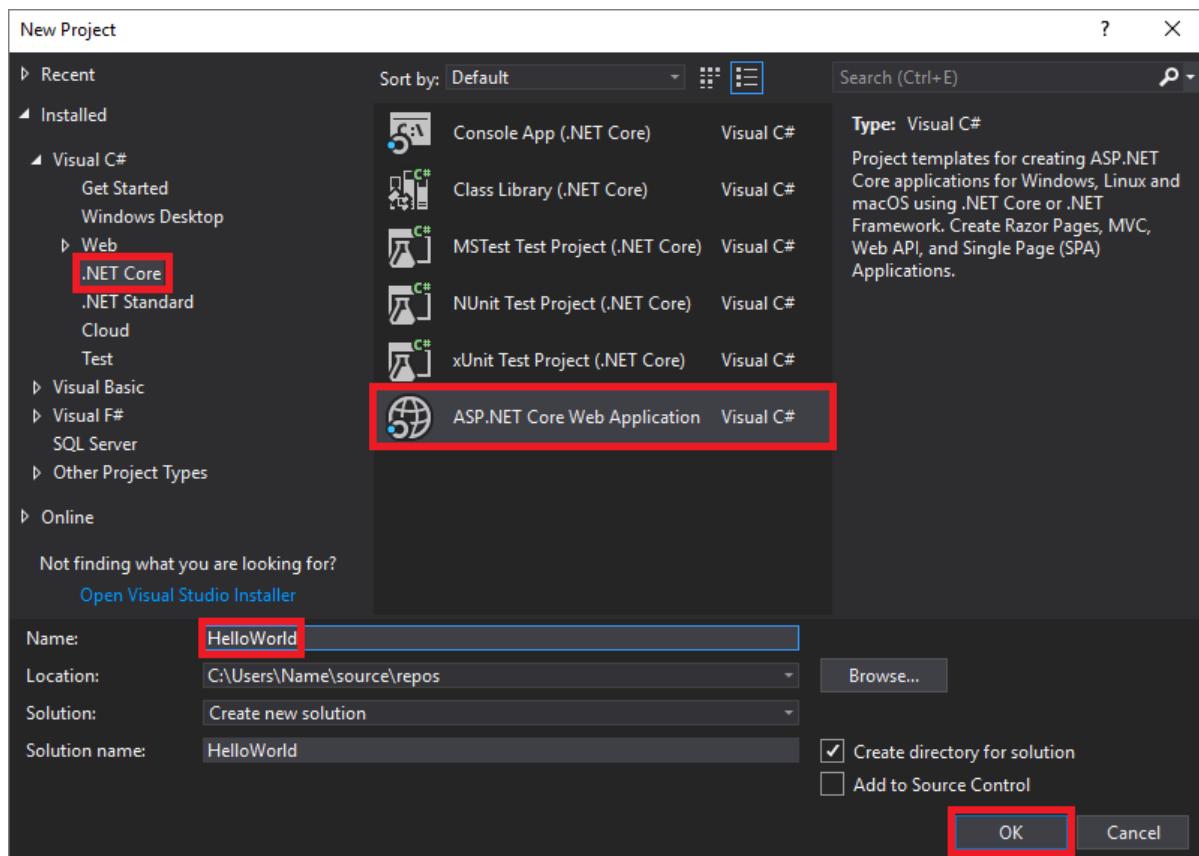
This quickstart tutorial includes screenshots that use the dark theme. If you aren't using the dark theme but would like to, see the [Personalize the Visual Studio IDE and Editor](#) page to learn how.

Create a project

To start, you'll create an ASP.NET Core web application project. The project type comes with all template files to create a web app, before you've even added anything!

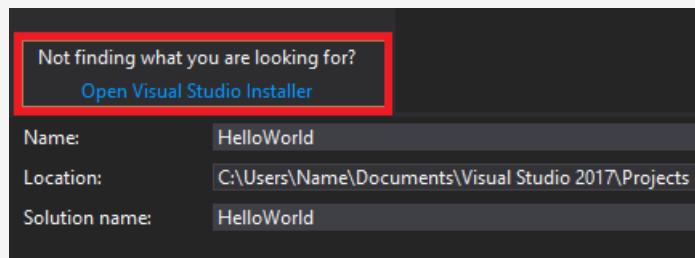
1. Open Visual Studio 2017.
2. From the top menu bar, choose **File > New > Project**.
3. In the left pane of the **New Project** dialog box, expand **Visual C#**, and then choose **.NET Core**. In the middle pane, choose **ASP.NET Core Web Application**.

Then, name your file `HelloWorld` and choose **OK**.

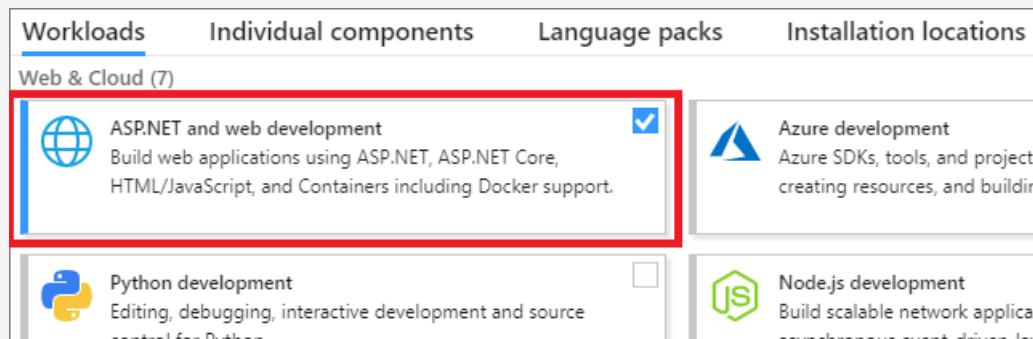


NOTE

If you don't see the **.NET Core** project template category, choose the **Open Visual Studio Installer** link in the left pane. (Depending on your display settings, you might have to scroll to see it.)

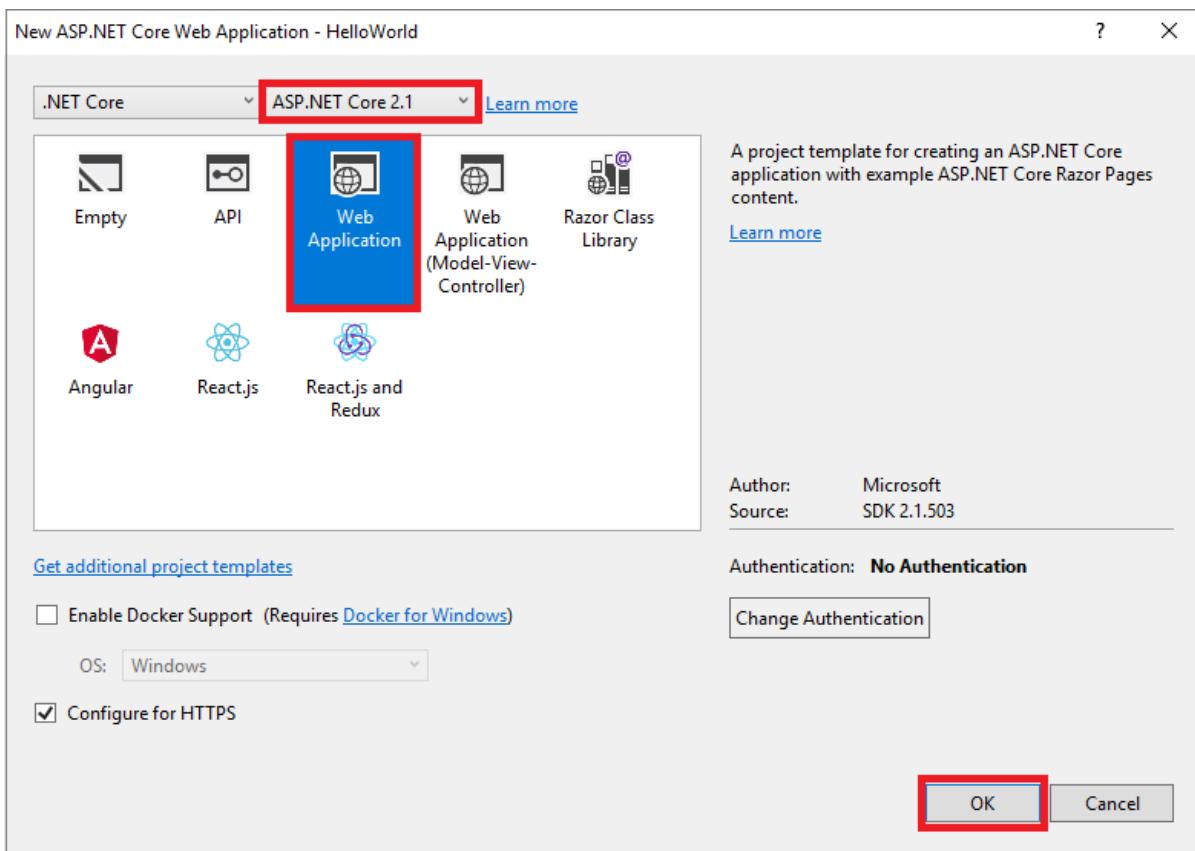


The Visual Studio Installer launches. Choose the **ASP.NET and web development** workload, and then choose **Modify**.



(You might have to close Visual Studio before you can continue installing the new workload.)

4. In the **New ASP.NET Core Web Application** dialog box, select **ASP.NET Core 2.1** from the top dropdown menu. Next, choose **Web Application**, and then choose **OK**.

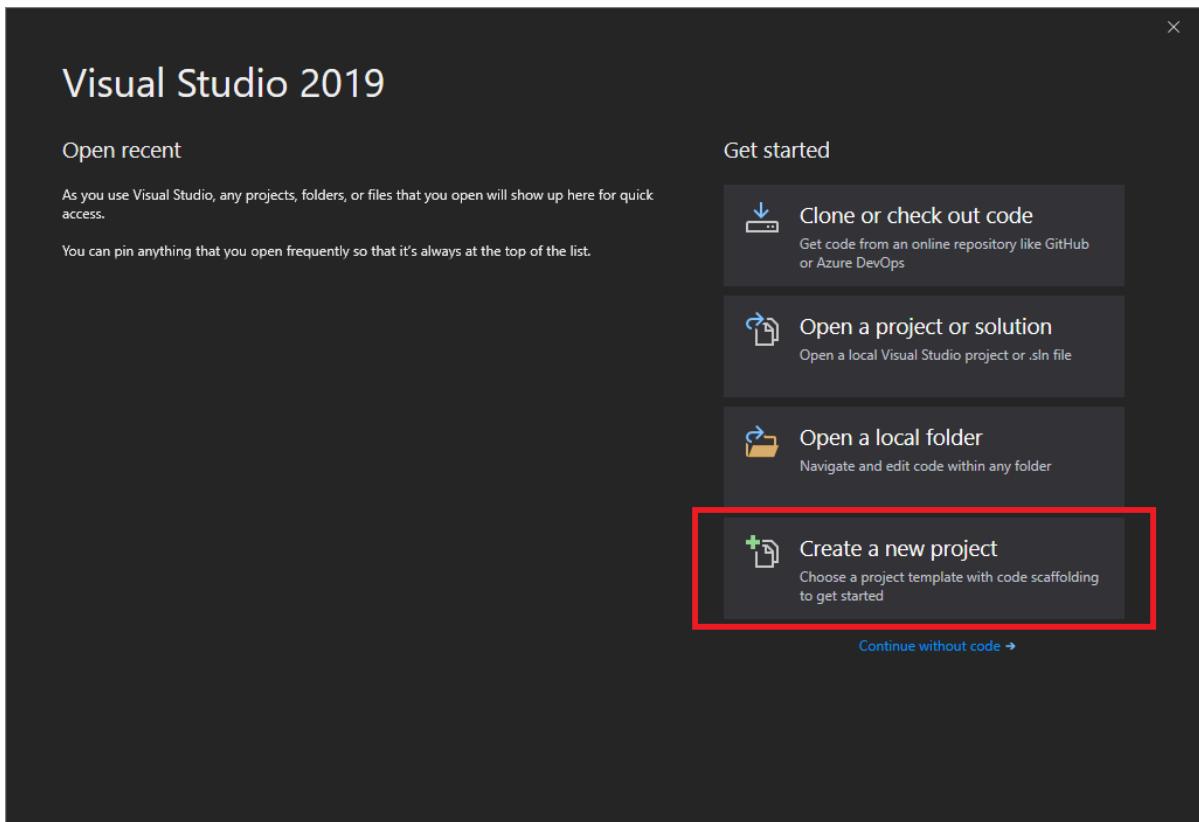


NOTE

If you don't see **ASP.NET Core 2.1**, make sure that you are running the most recent release of Visual Studio. For more information about how to update your installation, see the [Update Visual Studio to the most recent release page](#).

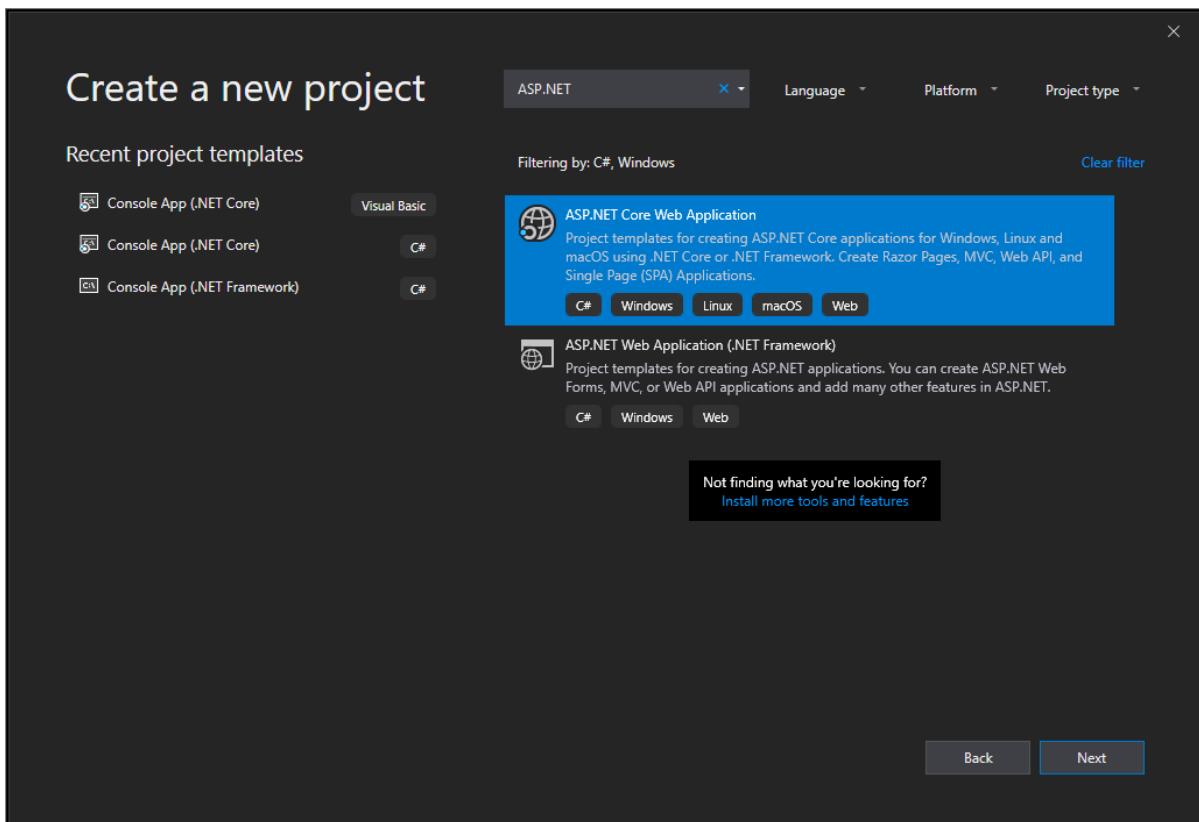
Soon after, Visual Studio opens your project file.

1. Open Visual Studio.
2. On the start window, choose **Create a new project**.



3. On the **Create a new project** window, enter or type **ASP.NET** in the search box. Next, choose **C#** from the Language list, and then choose **Windows** from the Platform list.

After you apply the language and platform filters, choose the **ASP.NET Core Web Application** template, and then choose **Next**.

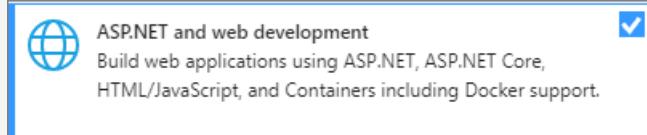


NOTE

If you do not see the **ASP.NET Core Web Application** template, you can install it from the **Create a new project** window. In the **Not finding what you're looking for?** message, choose the **Install more tools and features** link.

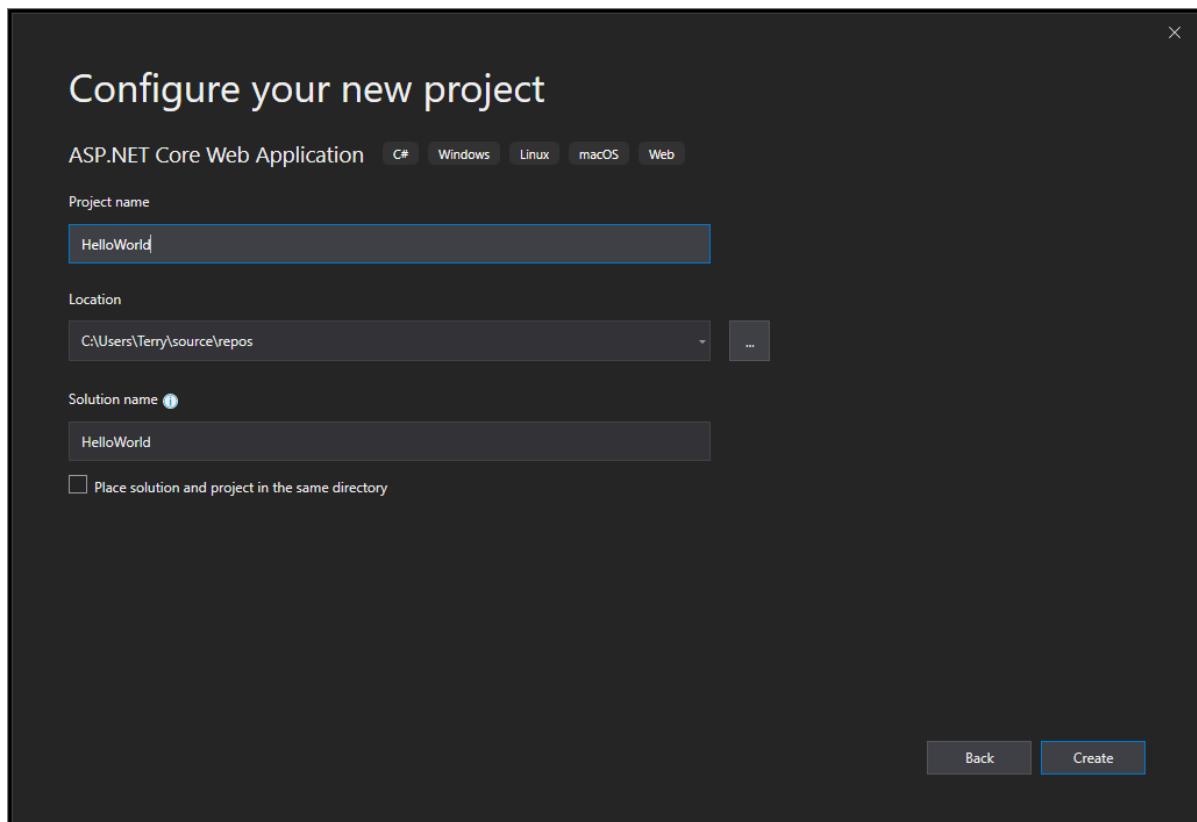
Not finding what you're looking for?
[Install more tools and features](#)

Then, in the Visual Studio Installer, choose the **ASP.NET and web development** workload.

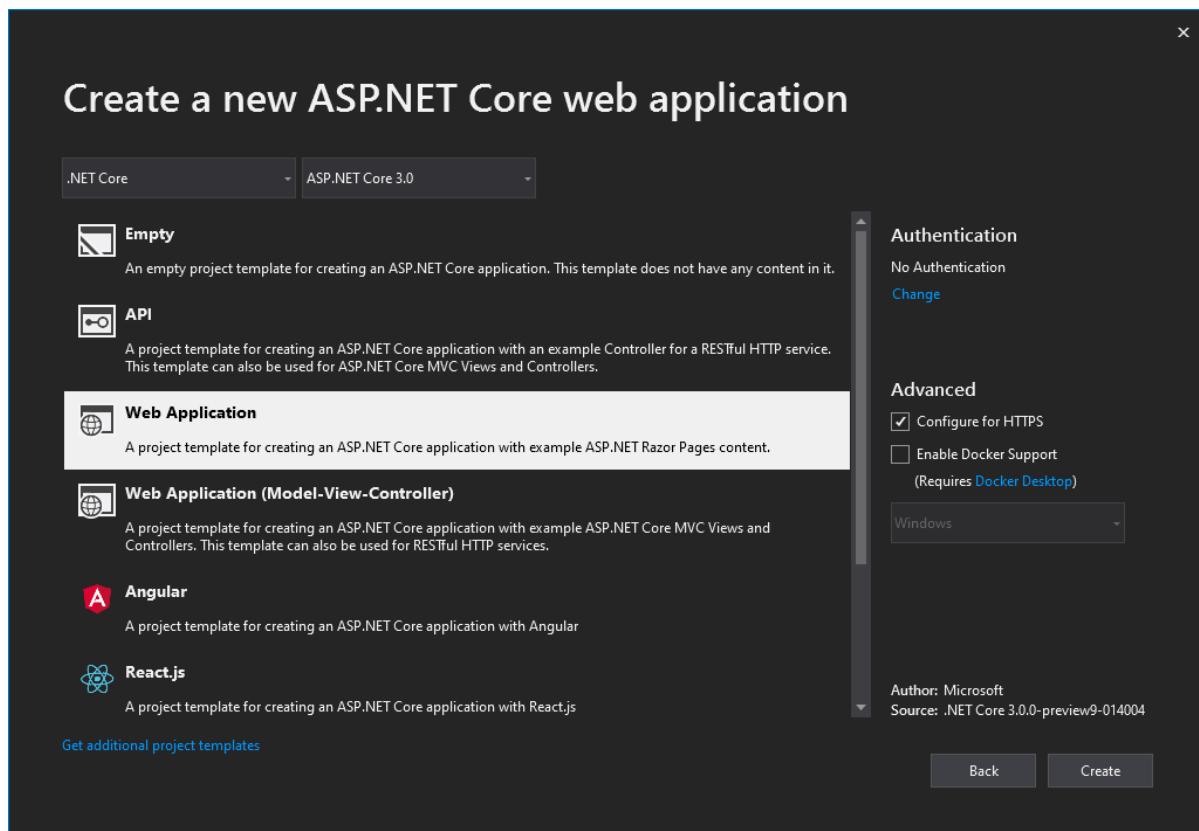


After that, choose the **Modify** button in the Visual Studio Installer. You might be prompted to save your work; if so, do so. Next, choose **Continue** to install the workload. Then, return to step 2 in this "Create a project" procedure.

4. In the **Configure your new project** window, type or enter *HelloWorld* in the **Project name** box. Then, choose **Create**.



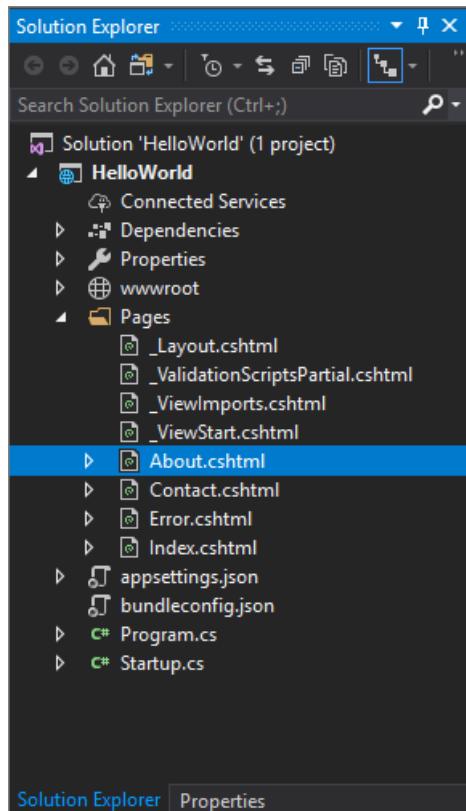
5. In the **Create a new ASP.NET Core Web Application** window, verify that **ASP.NET Core 3.0** appears in the top drop-down menu. Then, choose **Web Application**, which includes example Razor Pages. Next, choose **Create**.



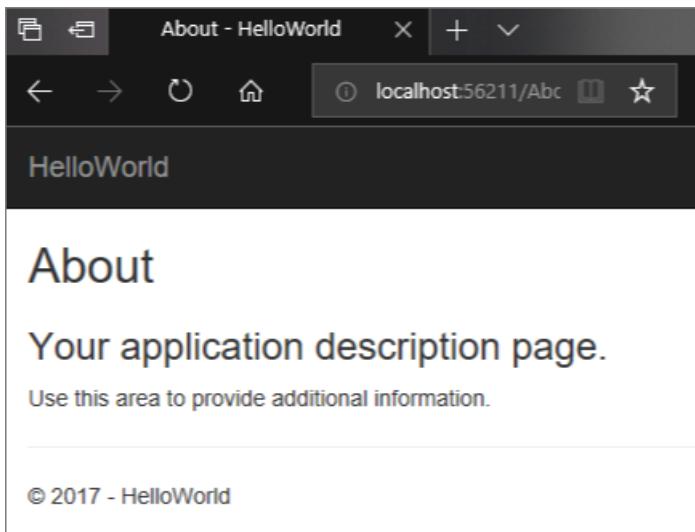
Visual Studio opens your new project.

Create and run the app

1. In the **Solution Explorer**, expand the **Pages** folder, and then choose **About.cshtml**.



This file corresponds to a page that's named **About** in the web app, which runs in a web browser.



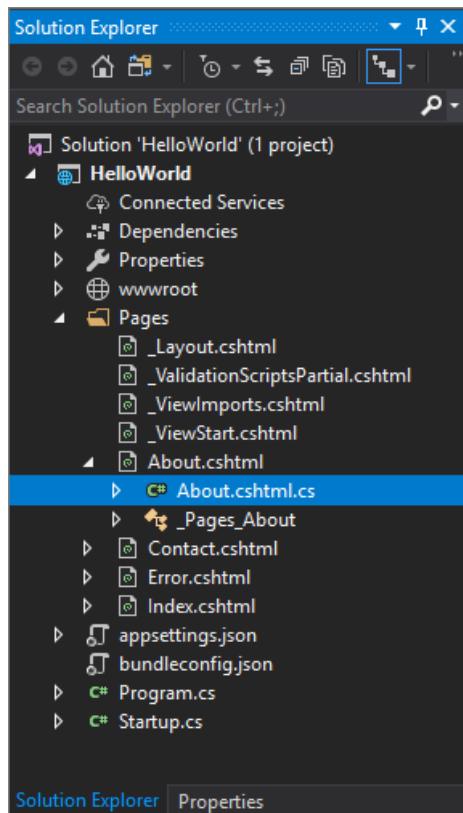
In the editor, you'll see HTML code for the "additional information" area of the **About** page.

```
 About.cshtml* ✘ X
 @page
 @model AboutModel
 @{
     ViewData["Title"] = "About";
 }
 <h2>@ViewData["Title"]</h2>
 <h3>@Model.Message</h3>
 <p>Use this area to provide additional information.</p>
```

2. Change the "additional information" text to read "**Hello World!**".

```
 About.cshtml* ✘ X
 @page
 @model AboutModel
 @{
     ViewData["Title"] = "About";
 }
 <h2>@ViewData["Title"]</h2>
 <h3>@Model.Message</h3>
 <p>Hello World!</p>
```

3. In the **Solution Explorer**, expand **About.cshtml**, and then choose **About.cshtml.cs**. (This file also corresponds with the **About** page in a web browser.)



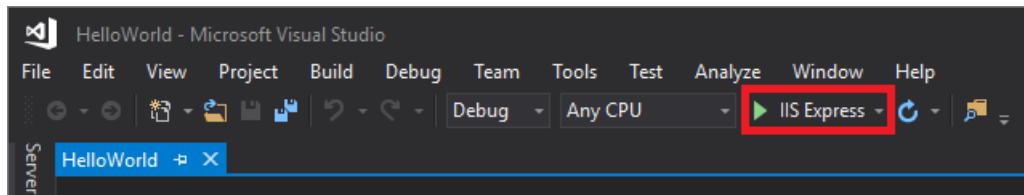
In the editor, you'll see C# code that includes text for the "application description" area of the **About** page.

```
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Threading.Tasks;
5  using Microsoft.AspNetCore.Mvc.RazorPages;
6
7  namespace HelloWorld.Pages
8  {
9      public class AboutModel : PageModel
10     {
11         public string Message { get; set; }
12
13         public void OnGet()
14         {
15             Message = "Your application description page.";
16         }
17     }
18 }
```

4. Change the "application description" message text to read "**What's my message?**".

```
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Threading.Tasks;
5  using Microsoft.AspNetCore.Mvc.RazorPages;
6
7  namespace HelloWorld.Pages
8  {
9      public class AboutModel : PageModel
10     {
11         public string Message { get; set; }
12
13         public void OnGet()
14         {
15             Message = "What's my message?";
16         }
17     }
18 }
```

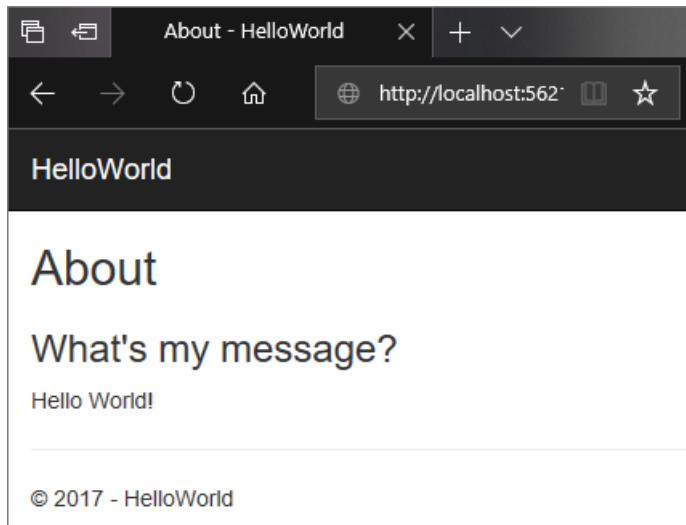
5. Choose **IIS Express** or press **Ctrl+F5** to run the app and open it in a web browser.



NOTE

If you get an error message that says, **Unable to connect to web server 'IIS Express'**, or an error message that mentions an SSL certificate, close Visual Studio. Next, open Visual Studio by using the **Run as administrator** option from the right-click context menu. Then, run the application again.

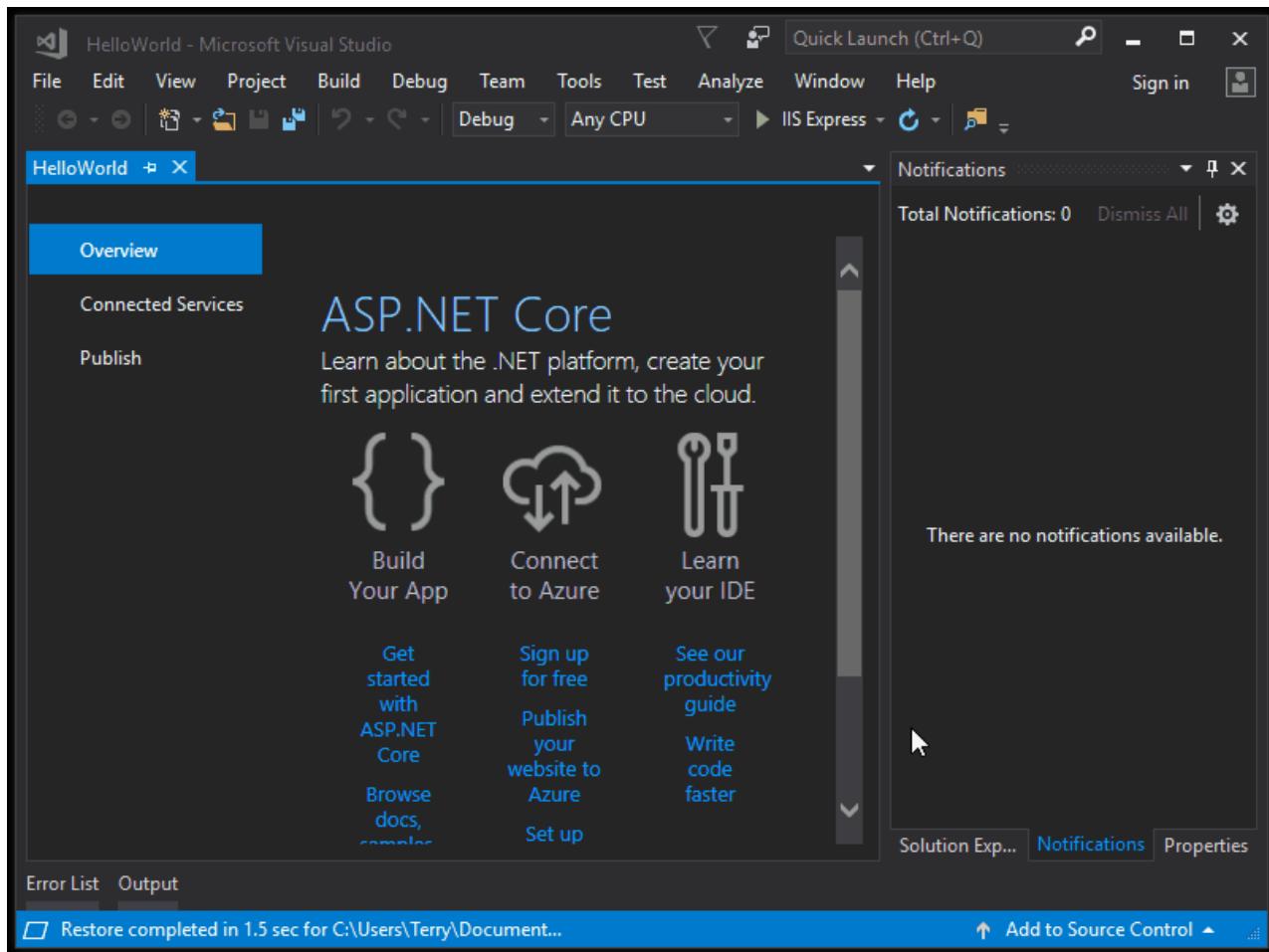
6. In the web browser, verify that the **About** page includes your updated text.



7. Close the web browser.

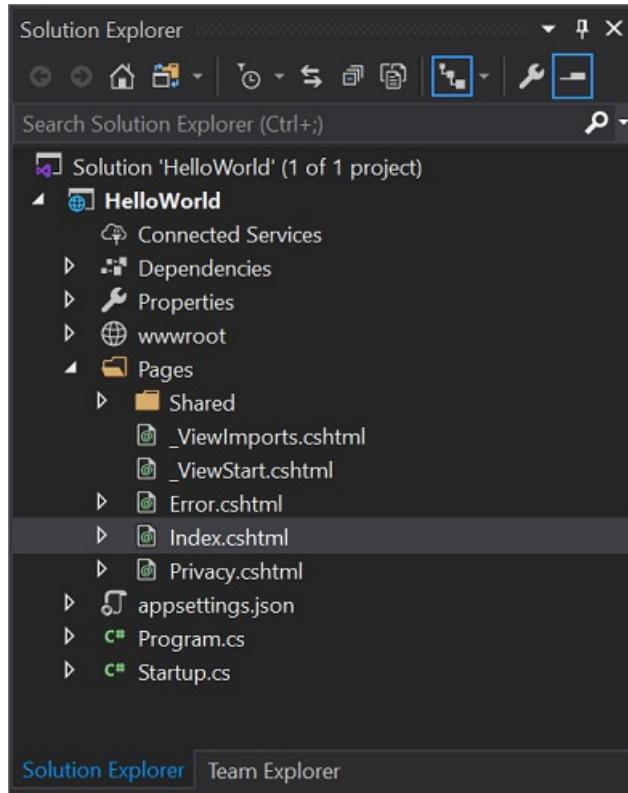
Review your work

View the following animation to check the work that you completed in the previous section.

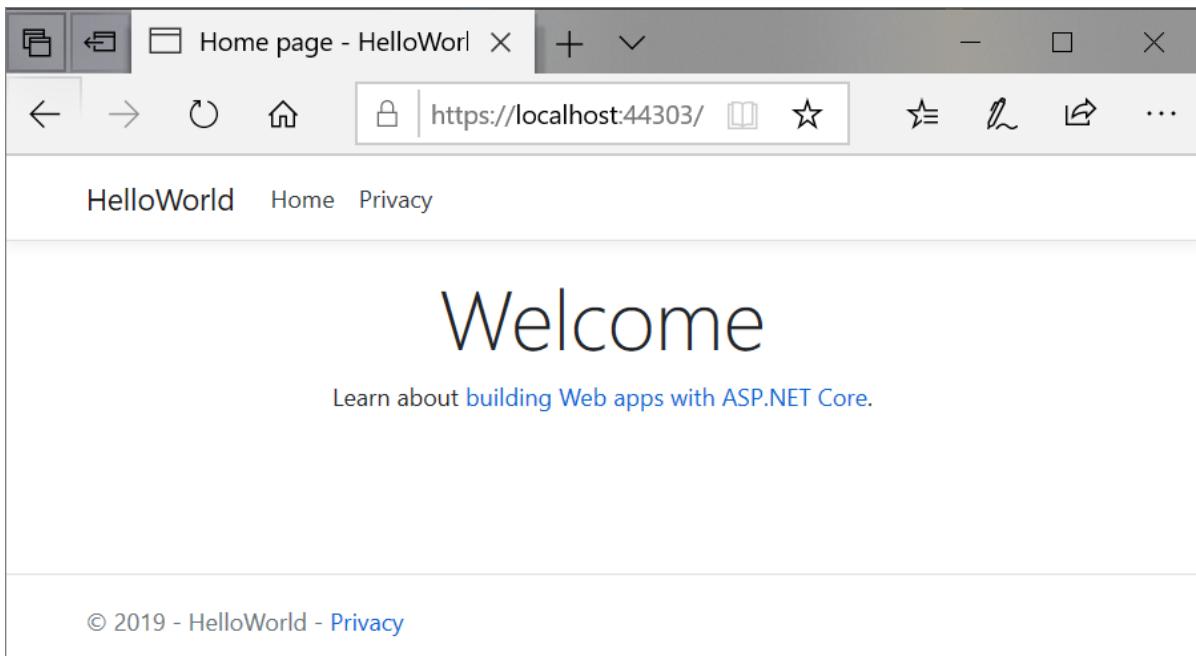


Congratulations on completing this Quickstart! We hope you learned a little bit about C#, ASP.NET Core, and the Visual Studio IDE (integrated development environment).

1. In the **Solution Explorer**, expand the **Pages** folder, and then choose **Index.cshtml**.



This file corresponds to a page that's named **Home** in the web app, which runs in a web browser.



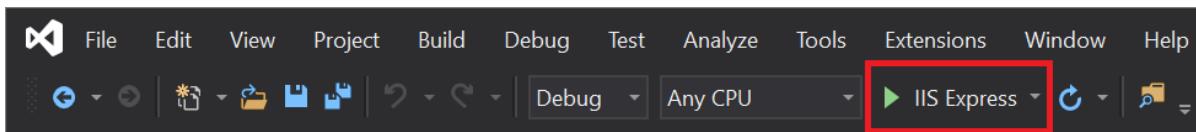
In the editor, you'll see HTML code for the text that appears on the **Home** page.

A screenshot of the Visual Studio code editor. The file is named "Index.cshtml". The code contains C# Razor syntax for setting ViewData["Title"] and displaying a welcome message with a link to the ASP.NET Core documentation.

2. Change the "Welcome" text to read "**Hello World!**".

A screenshot of the Visual Studio code editor showing the same "Index.cshtml" file. The "Welcome" text has been replaced by "Hello World!". The word "Hello" is highlighted with a red rectangular box.

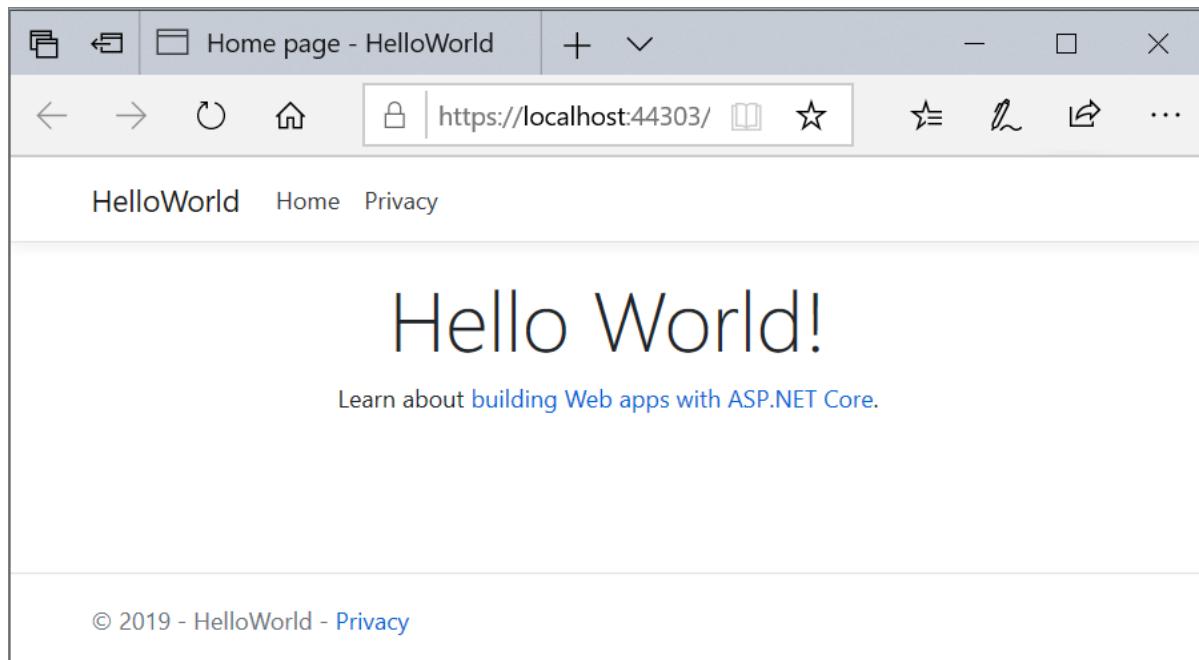
3. Choose **IIS Express** or press **Ctrl+F5** to run the app and open it in a web browser.



NOTE

If you get an error message that says, **Unable to connect to web server 'IIS Express'**, or an error message that mentions an SSL certificate, close Visual Studio. Next, open Visual Studio by using the **Run as administrator** option from the right-click context menu. Then, run the application again.

4. In the web browser, verify that the **Home** page includes your updated text.



5. Close the web browser.

Next steps

To learn more, continue with the following tutorial:

[Get started with C# and ASP.NET in Visual Studio](#)

See also

[Publish your web app to Azure App Service by using Visual Studio](#)

Quickstart: Use Visual Studio to create your first C# console app

8/30/2019 • 2 minutes to read • [Edit Online](#)

In this 5-10 minute introduction to the Visual Studio integrated development environment (IDE), you'll create a simple C# app that runs on the console.

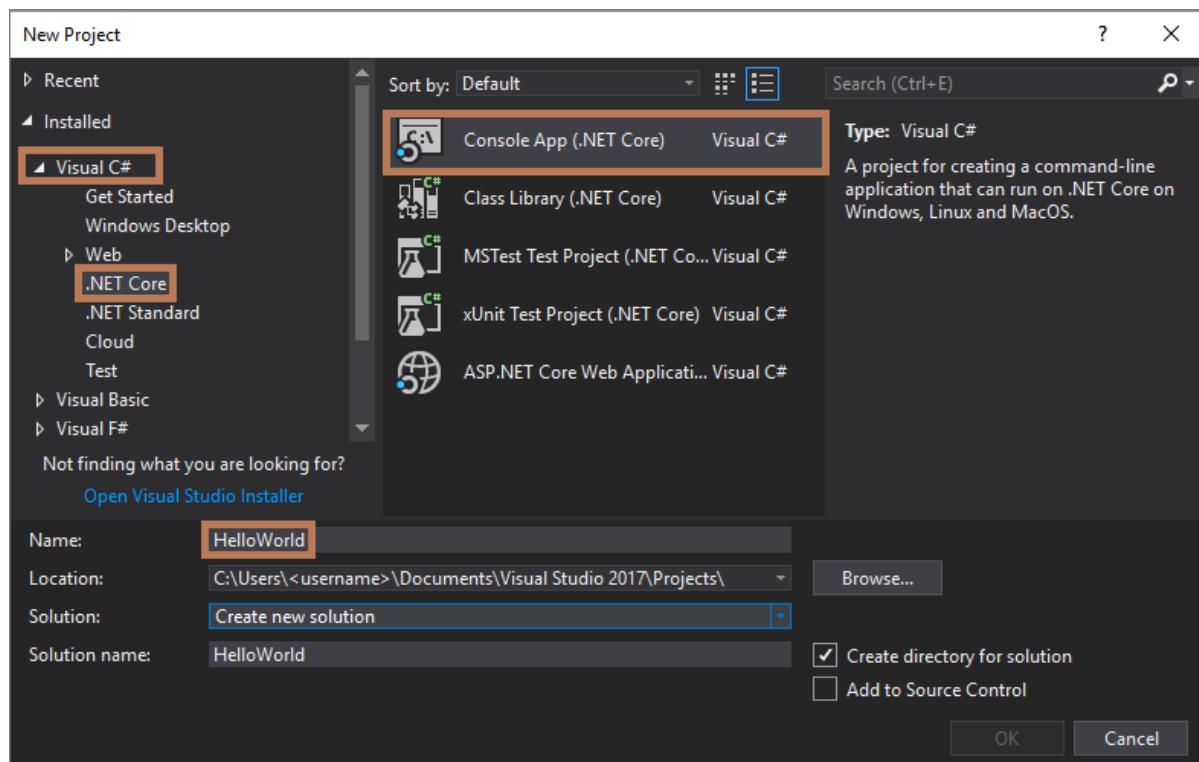
If you haven't already installed Visual Studio, go to the [Visual Studio downloads](#) page to install it for free.

If you haven't already installed Visual Studio, go to the [Visual Studio downloads](#) page to install it for free.

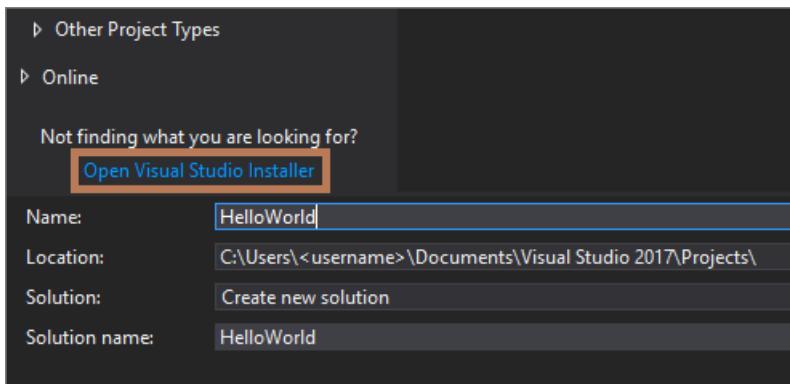
Create a project

First, you'll create a C# application project. The project type comes with all the template files you'll need, before you've even added anything!

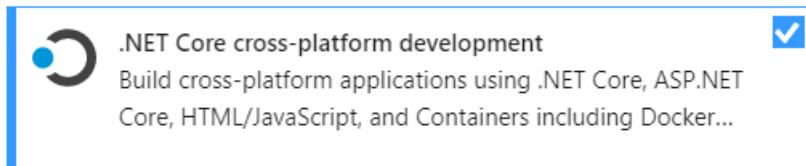
1. Open Visual Studio 2017.
2. From the top menu bar, choose **File > New > Project**.
3. In the **New Project** dialog box in the left pane, expand **C#**, and then choose **.NET Core**. In the middle pane, choose **Console App (.NET Core)**. Then name the project *HelloWorld*.



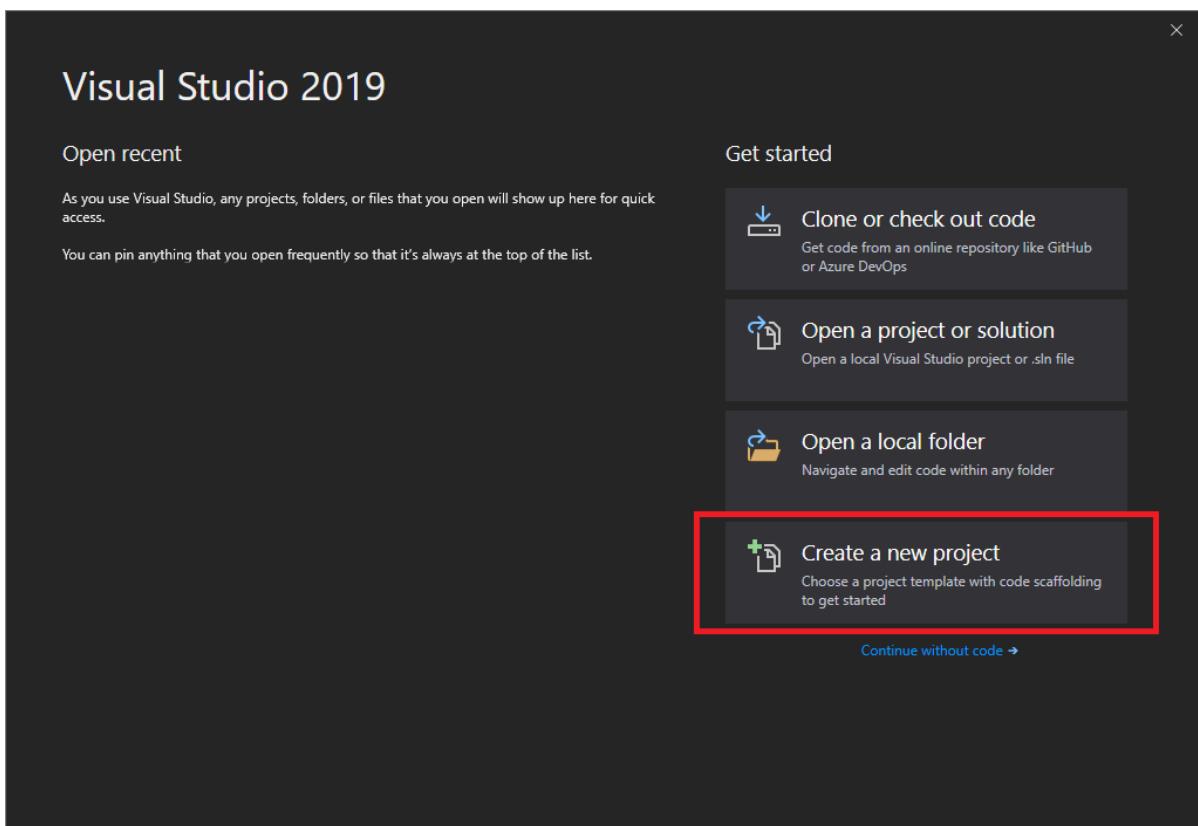
If you don't see the **Console App (.NET Core)** project template, choose the [Open Visual Studio Installer](#) link in the left pane of the **New Project** dialog box.



The Visual Studio Installer launches. Choose the **.NET Core cross-platform development** workload, and then choose **Modify**.

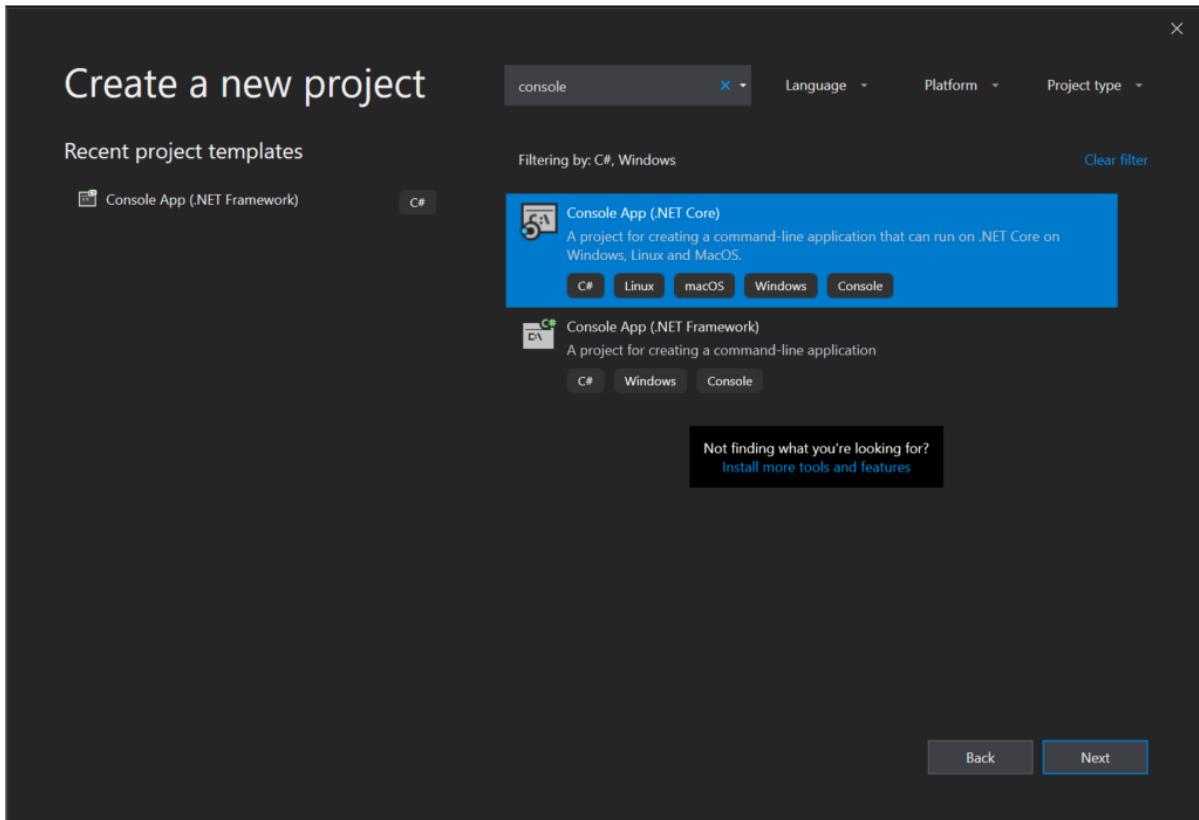


1. Open Visual Studio 2019.
2. On the start window, choose **Create a new project**.



3. On the **Create a new project** window, enter or type *console* in the search box. Next, choose **C#** from the Language list, and then choose **Windows** from the Platform list.

After you apply the language and platform filters, choose the **Console App (.NET Core)** template, and then choose **Next**.

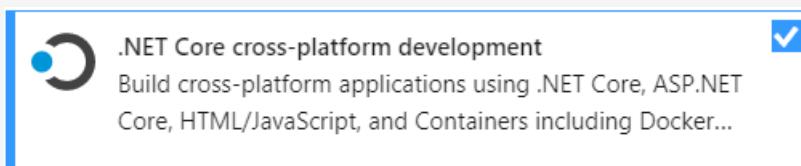


NOTE

If you do not see the **Console App (.NET Core)** template, you can install it from the **Create a new project** window. In the **Not finding what you're looking for?** message, choose the **Install more tools and features** link.

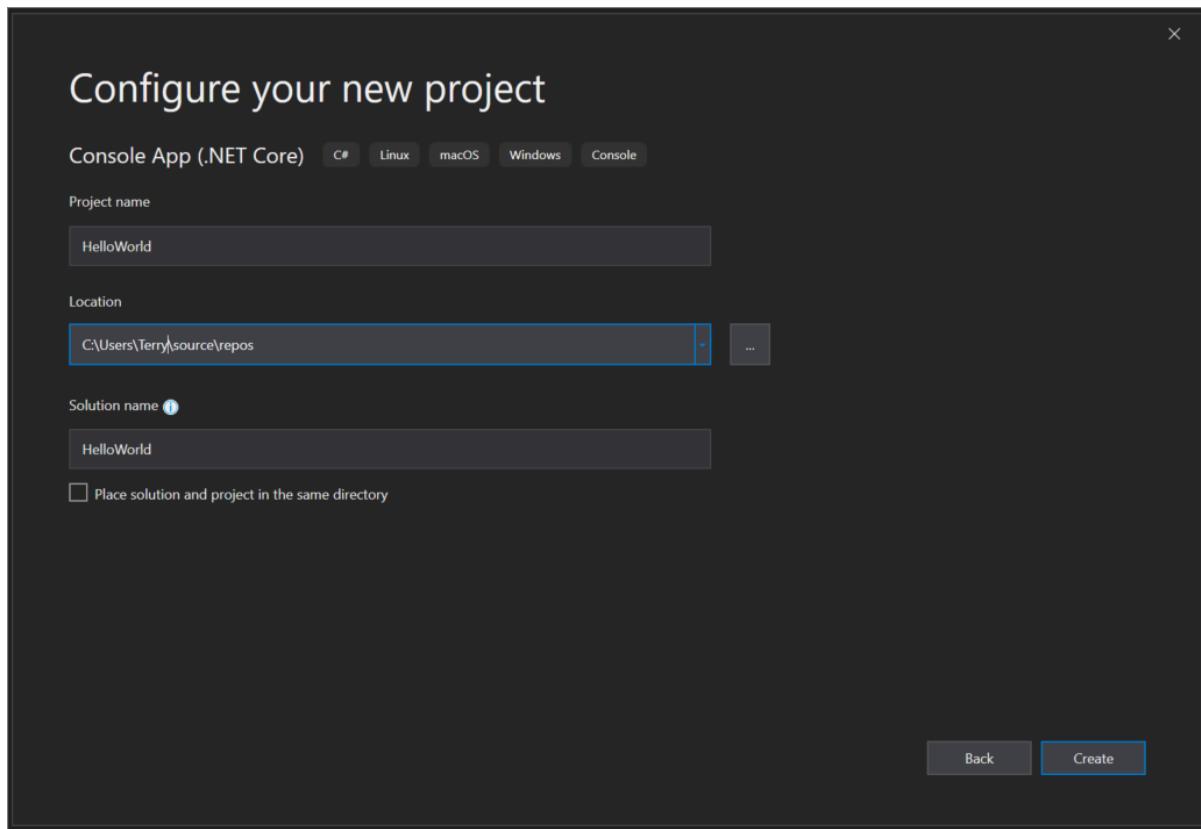
Not finding what you're looking for?
[Install more tools and features](#)

Then, in the Visual Studio Installer, choose the **.NET Core cross-platform development** workload.



After that, choose the **Modify** button in the Visual Studio Installer. You might be prompted to save your work; if so, do so. Next, choose **Continue** to install the workload. Then, return to step 2 in this "Create a project" procedure.

4. In the **Configure your new project** window, type or enter *HelloWorld* in the **Project name** box. Then, choose **Create**.



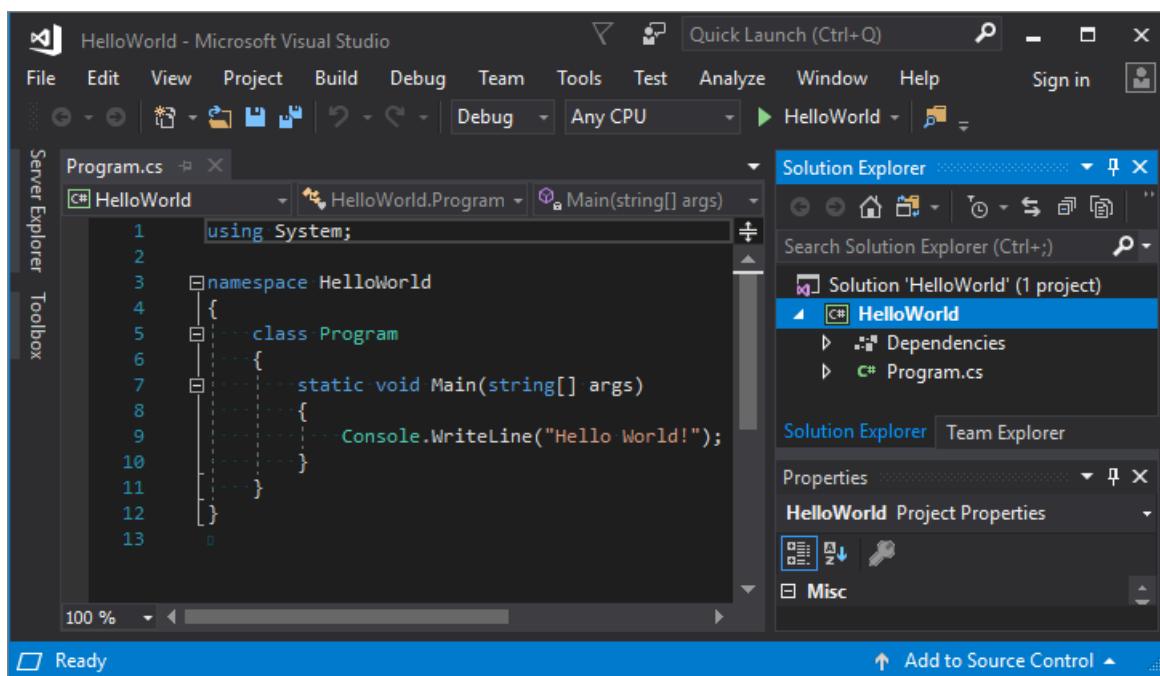
Visual Studio opens your new project.

Create the application

After you select your C# project template and name your project, Visual Studio creates a simple "Hello World" application for you.

Visual Studio includes default "Hello World" code in your project.

(To do so, it calls the [WriteLine](#) method to display the literal string "Hello World!" in the console window.)



If you press **F5**, you can run the program in Debug mode. However, the console window is visible only for a moment before it closes.

(This behavior happens because the `Main` method terminates after its single statement executes, and so the

application ends.)

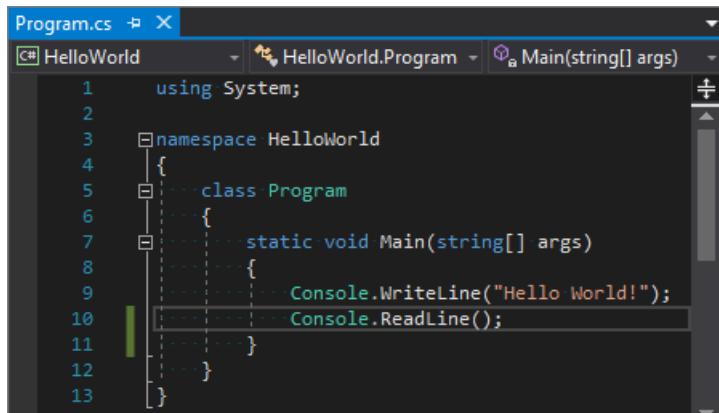
Add some code

Let's add some code to pause the application so that the console window doesn't close until you press **ENTER**.

1. Add the following code immediately after the call to the [WriteLine](#) method:

```
Console.ReadLine();
```

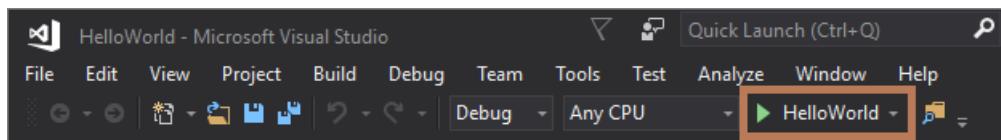
2. Verify that it looks like this in the code editor:



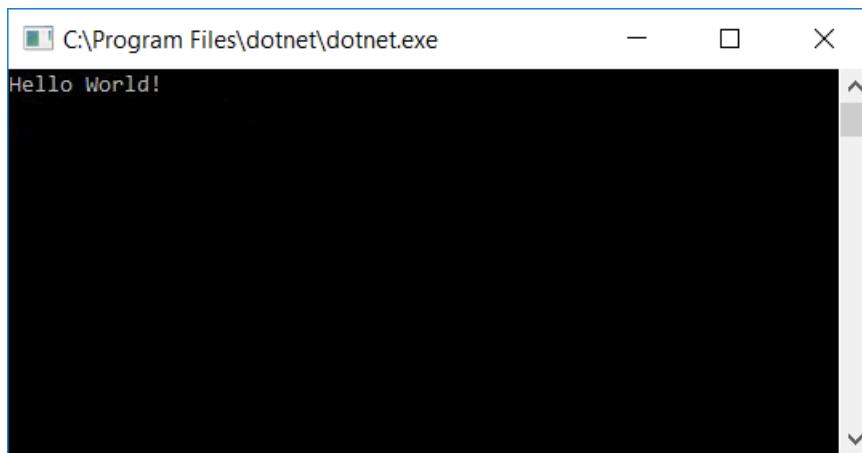
```
1  using System;
2
3  namespace HelloWorld
4  {
5      class Program
6      {
7          static void Main(string[] args)
8          {
9              Console.WriteLine("Hello World!");
10             Console.ReadLine();
11         }
12     }
13 }
```

Run the application

1. Choose the **HelloWorld** button on the toolbar to run the application in Debug mode. (Or, you can press **F5**.)

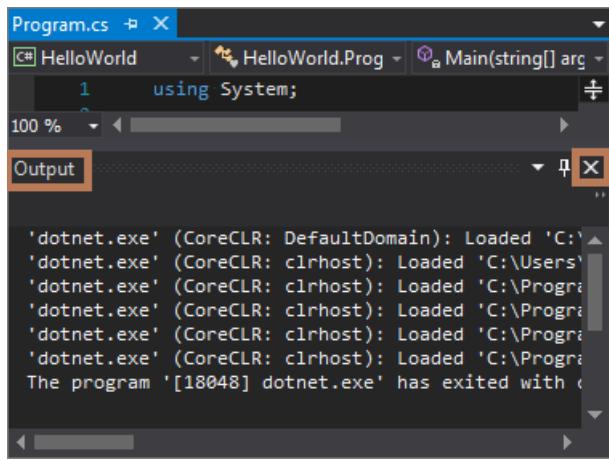


2. View your app in the console window.



Close the application

1. Press **ENTER** to close the console window.
2. Close the **Output** pane in Visual Studio.



A screenshot of the Visual Studio IDE. The top menu bar shows 'Program.cs' and 'HelloWorld'. Below the menu is a code editor with a single line of C# code: 'using System;'. The 'Output' window is open at the bottom, showing the following log entries:

```
'dotnet.exe' (CoreCLR: DefaultDomain): Loaded 'C:\Windows\Microsoft.NET\assembly\GAC_MSIL\System\v4.0_4.0.0.0__b77a5c561934e089\System.dll'  
'dotnet.exe' (CoreCLR: clrhost): Loaded 'C:\Users\\AppData\Local\Temp\dotnetapp1111\dotnetapp1111.dll'  
'dotnet.exe' (CoreCLR: clrhost): Loaded 'C:\Windows\Microsoft.NET\assembly\GAC_MSIL\System\v4.0_4.0.0.0__b77a5c561934e089\System.dll'  
'dotnet.exe' (CoreCLR: clrhost): Loaded 'C:\Windows\Microsoft.NET\assembly\GAC_MSIL\System.Core\v4.0_4.0.0.0__b77a5c561934e089\System.Core.dll'  
'dotnet.exe' (CoreCLR: clrhost): Loaded 'C:\Windows\Microsoft.NET\assembly\GAC_MSIL\System.Numerics\v4.0_4.0.0.0__b77a5c561934e089\System.Numerics.dll'  
The program '[18048] dotnet.exe' has exited with code 0.
```

3. Close Visual Studio.

Next steps

Congratulations on completing this Quickstart! We hope you learned a little bit about C# and the Visual Studio IDE. To learn more, continue with the following tutorials.

[Get started with a C# console app in Visual Studio](#)

Quickstart: Create your first console app in Visual Studio with Visual Basic

8/30/2019 • 3 minutes to read • [Edit Online](#)

In this 5-10 minute introduction to the Visual Studio integrated development environment (IDE), you'll create a simple Visual Basic application that runs on the console.

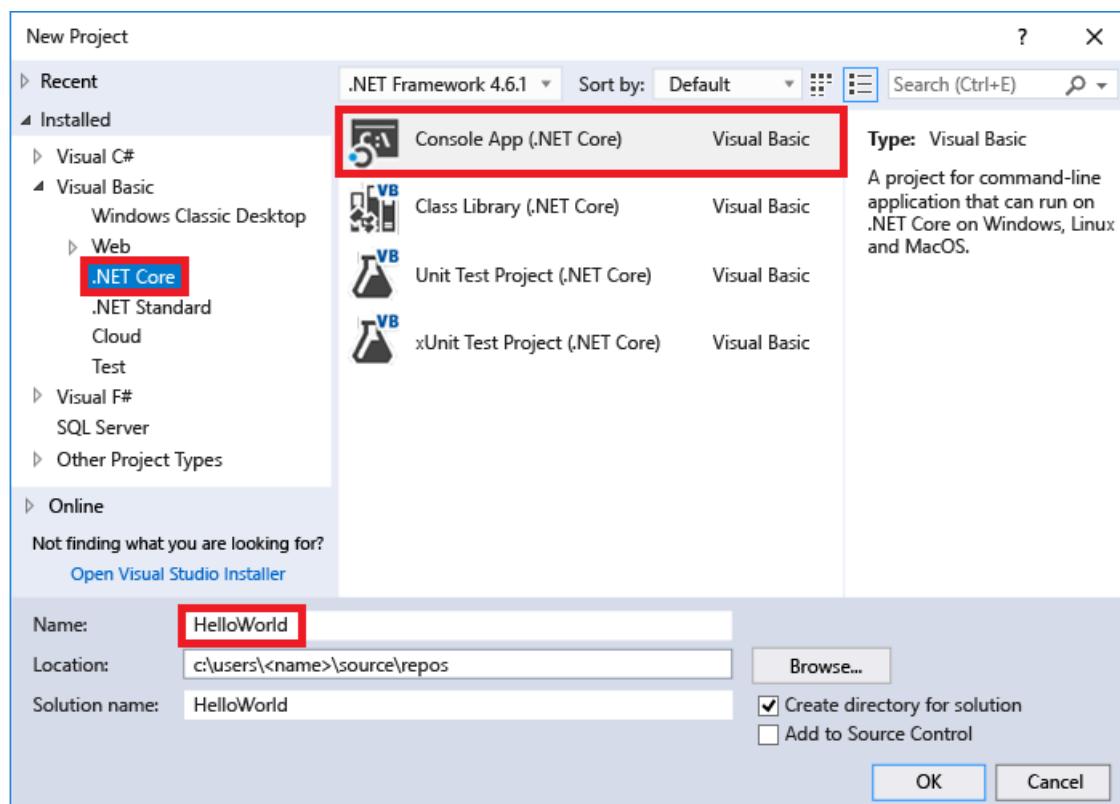
If you haven't already installed Visual Studio, go to the [Visual Studio downloads](#) page to install it for free.

If you haven't already installed Visual Studio, go to the [Visual Studio downloads](#) page to install it for free.

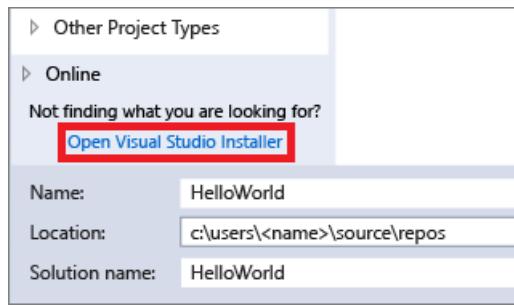
Create a project

First, you'll create a Visual Basic application project. The project type comes with all the template files you'll need, before you've even added anything!

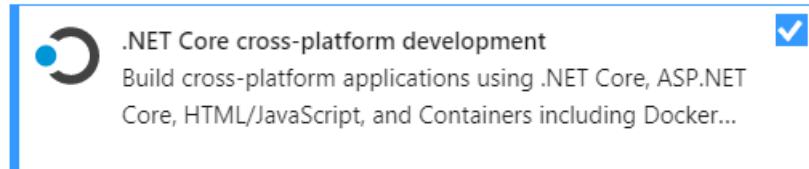
1. Open Visual Studio 2017.
2. From the top menu bar, choose **File > New > Project**.
3. In the **New Project** dialog box in the left pane, expand **Visual Basic**, and then choose **.NET Core**. In the middle pane, choose **Console App (.NET Core)**. Then name the project *HelloWorld*.



If you don't see the **Console App (.NET Core)** project template, click the [Open Visual Studio Installer](#) link in the left pane of the **New Project** dialog box.



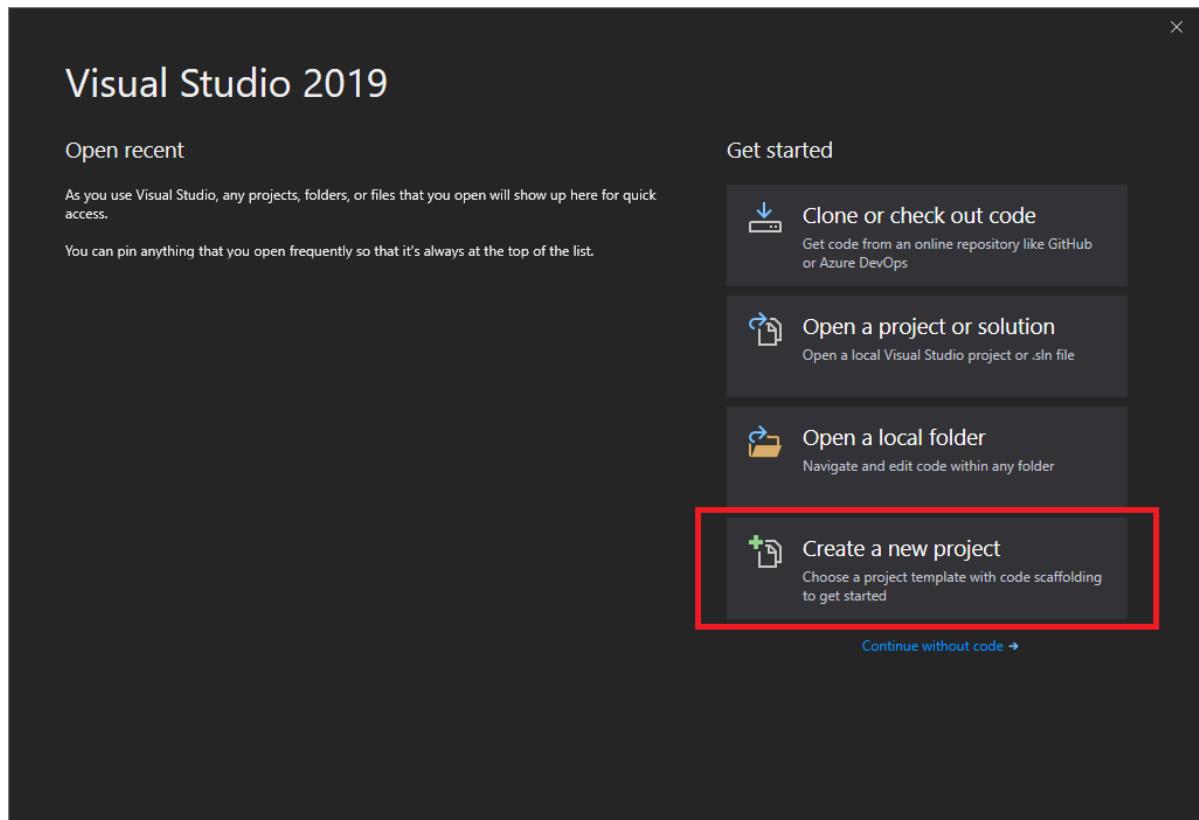
The Visual Studio Installer launches. Choose the **.NET Core cross-platform development** workload, and then choose **Modify**.



NOTE

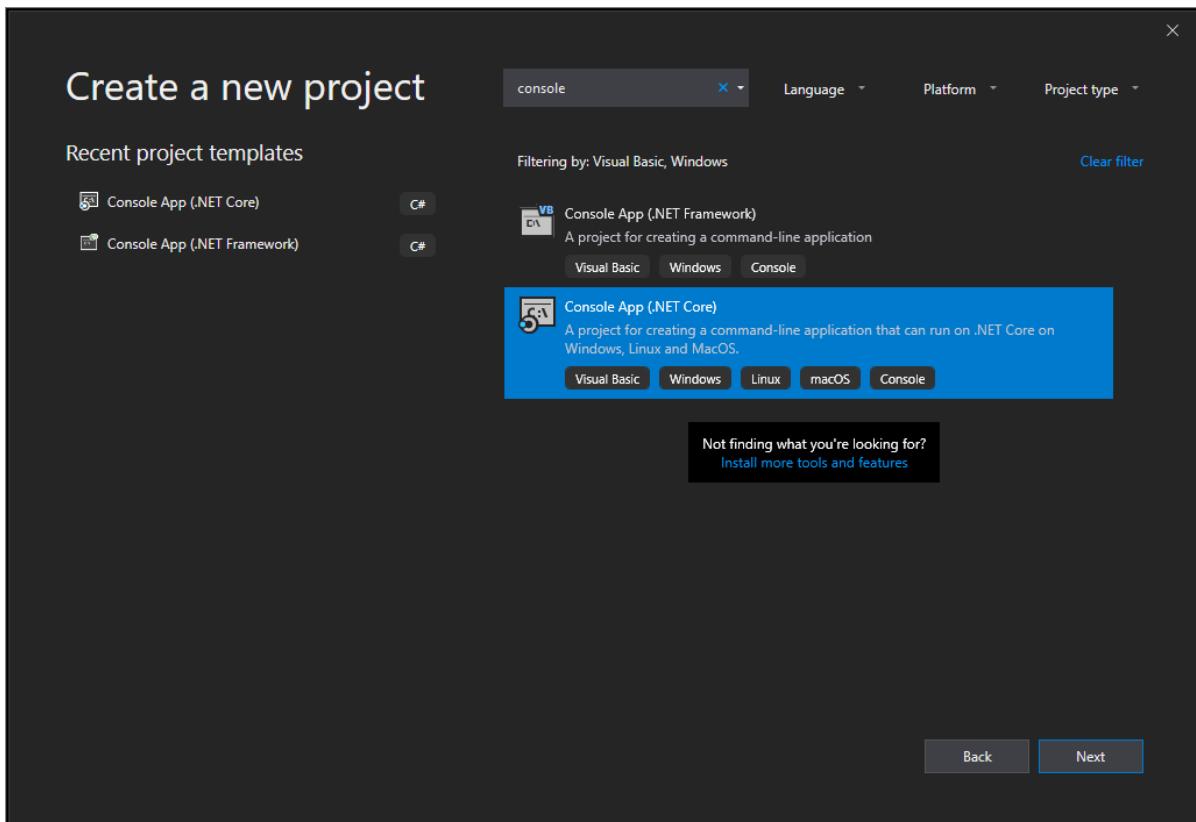
Some of the screenshots in this Quickstart use the dark theme. If you aren't using the dark theme but would like to, see the [Personalize the Visual Studio IDE and Editor](#) page to learn how.

1. Open Visual Studio 2019.
2. On the start window, choose **Create a new project**.



3. On the **Create a new project** window, enter or type *console* in the search box. Next, choose **Visual Basic** from the Language list, and then choose **Windows** from the Platform list.

After you apply the language and platform filters, choose the **Console App (.NET Core)** template, and then choose **Next**.

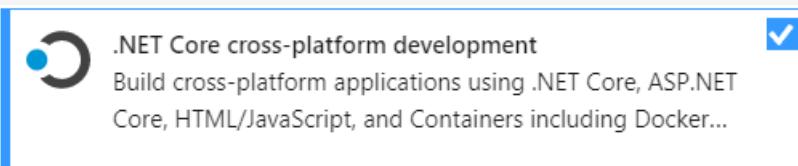


NOTE

If you do not see the **Console App (.NET Core)** template, you can install it from the **Create a new project** window. In the **Not finding what you're looking for?** message, choose the **Install more tools and features** link.

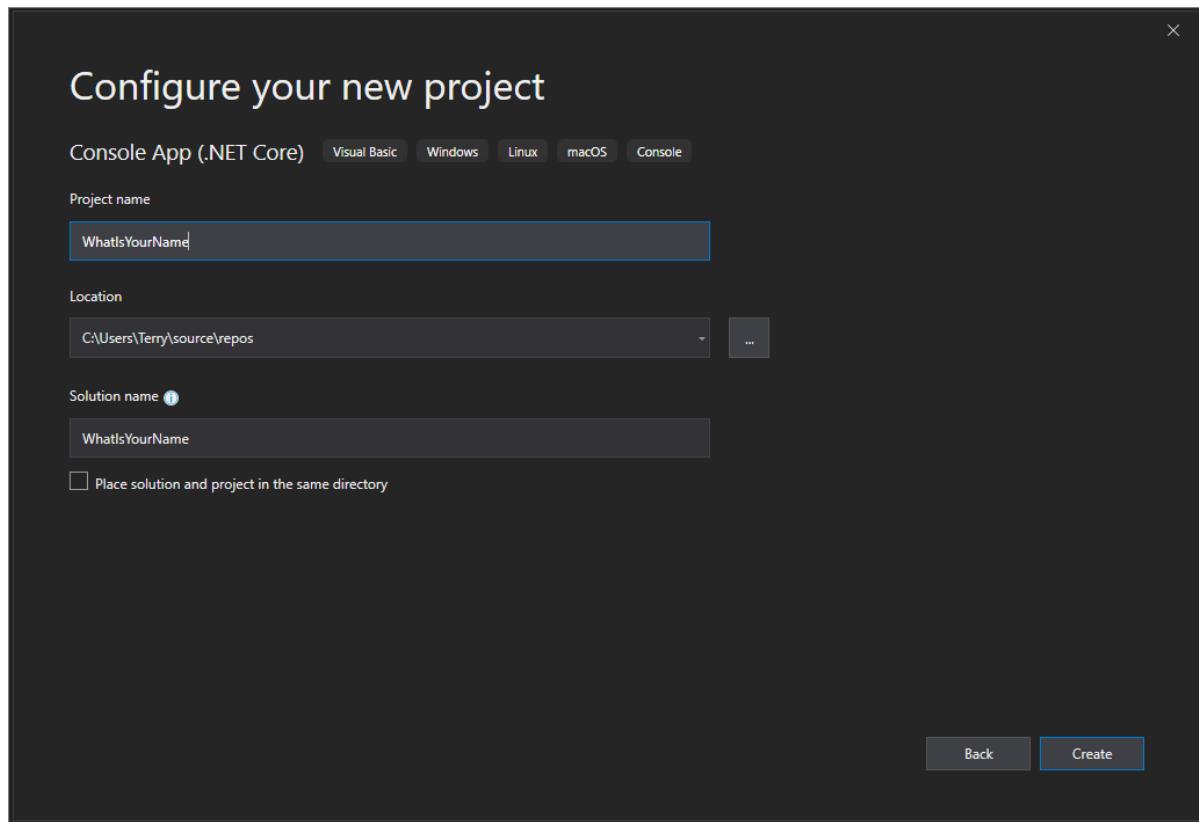
Not finding what you're looking for?
[Install more tools and features](#)

Then, in the Visual Studio Installer, choose the **.NET Core cross-platform development** workload.



After that, choose the **Modify** button in the Visual Studio Installer. You might be prompted to save your work; if so, do so. Next, choose **Continue** to install the workload. Then, return to step 2 in this "Create a project" procedure.

4. In the **Configure your new project** window, type or enter *WhatIsYourName* in the **Project name** box. Then, choose **Create**.



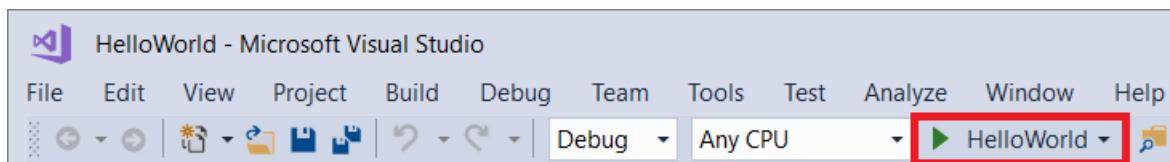
Visual Studio opens your new project.

Create the application

After you select your Visual Basic project template and name your project, Visual Studio creates a simple "Hello World" application for you. It calls the `WriteLine` method to display the literal string "Hello World!" in the console window.

```
Program.vb  X
VB HelloWorld  Program
1 Imports System
2
3 Module Program
4     Sub Main(args As String())
5         Console.WriteLine("Hello World!")
6     End Sub
7 End Module
8
```

If you click the **HelloWorld** button in the IDE, you can run the program in Debug mode.



When you do this, the console window is visible for only a moment before it closes. This happens because the `Main` method terminates after its single statement executes, and so the application ends.

Add some code

Let's add some code to pause the application and then ask for user input.

1. Add the following code immediately after the call to the `WriteLine` method:

```
Console.WriteLine("Press any key to continue...")
Console.ReadKey(true)
```

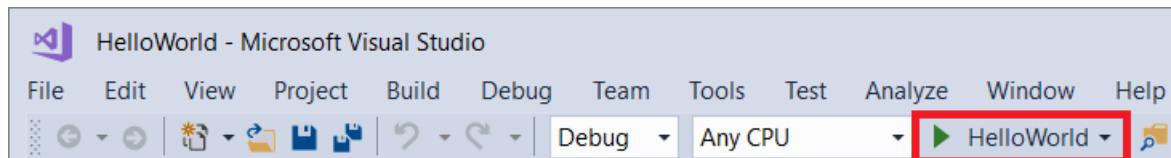
This pauses the program until you press a key.

2. On the menu bar, select **Build > Build Solution**.

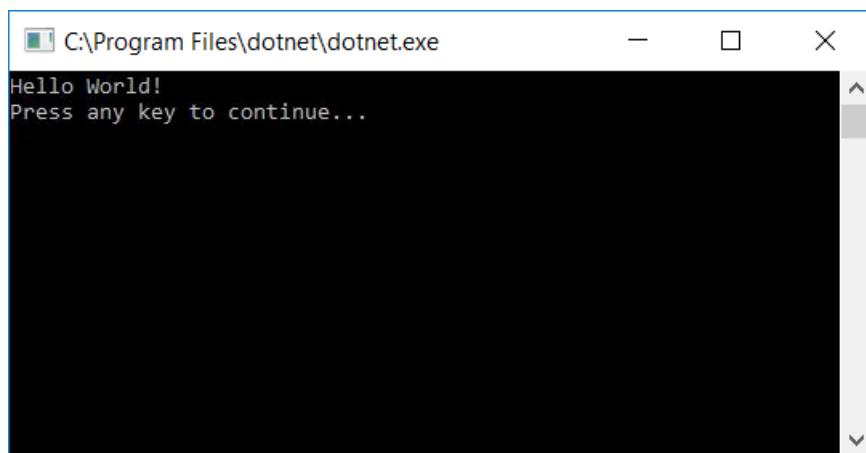
This compiles your program into an intermediate language (IL) that's converted into binary code by a just-in-time (JIT) compiler.

Run the application

1. Click the **HelloWorld** button on the toolbar.



2. Press any key to close the console window.



Next steps

Congratulations on completing this Quickstart! We hope you learned a little bit about Visual Basic and the Visual Studio IDE. To learn more, continue with the following tutorial.

[Get started with Visual Basic in Visual Studio](#)

How to: Move around in the Visual Studio IDE

10/18/2019 • 4 minutes to read • [Edit Online](#)

The integrated development environment (IDE) has been designed to allow you to move from window to window and file to file in several different ways, depending on your preference or project requirements. You can choose to cycle through open files in the editor, or cycle through all active tool windows in the IDE. You also can switch directly to any file open in the editor, regardless of the order in which it was last accessed. These features can help increase your productivity when working in the IDE.

NOTE

The options available in dialog boxes, and the names and locations of menu commands you see, might differ from what is described in this article, depending on your active settings or edition. This article was written with **General** settings in mind. To change your settings, for example to **General** or **Visual C++** settings, choose **Tools > Import and Export Settings**, and then choose **Reset all settings**.

Keyboard shortcuts

Almost every menu command in Visual Studio has a keyboard shortcut. You can also create your own custom shortcuts. For more information, see [Identify and customize keyboard shortcuts](#).

Navigate among files in the editor

You can use several methods to move through the files open in the editor. You can move among files based on the order in which you access them, use the IDE Navigator to quickly find any file currently open, or pin favorite files to the tab well so that they are always visible.

Navigate backward and navigate forward cycle through the open files in the editor based on the order in which they were accessed, much like back and forward do for your viewing history in Microsoft Internet Explorer.

To move through open files in order of use

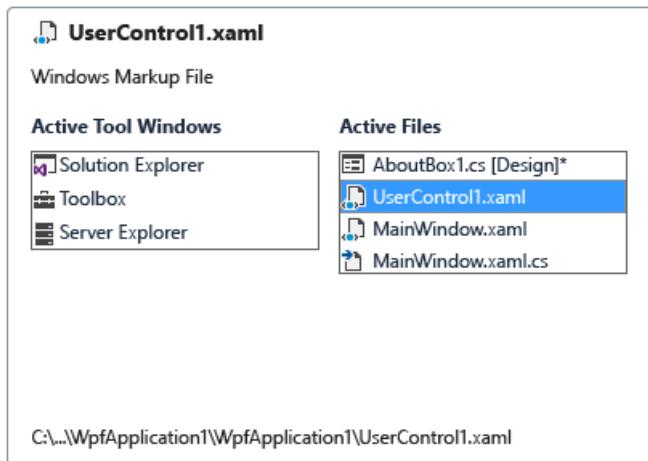
- To activate open documents in the order they were most recently touched, press **Ctrl+-** (hyphen).
- To activate open documents in the reverse order, press **Ctrl+Shift+-** (hyphen).

NOTE

Navigate Backward and **Navigate Forward** also can be found on the **View** menu.

You also can switch to a specific file open in the editor, regardless of when you last accessed the file, using the **IDE Navigator**, the **Active Files** list in the editor, or the **Windows** dialog box.

The **IDE Navigator** works much like the Windows application switcher. It is not available from menus and can be accessed only using shortcut keys. You can use either of two commands to access the **IDE Navigator** (shown below) to cycle through files, depending on the order in which you want to cycle through.



`Window.PreviousDocumentWindowNav` allows you to move to the file most recently accessed and `Window.NextDocumentWindowNav` allows you to move in the reverse order. **General Development Settings** assigns **Shift+Alt+F7** to `Window.PreviousDocumentWindowNav` and **Alt+F7** to `Window.NextDocumentWindowNav`.

NOTE

If the settings combination you are using does not already have a shortcut key combination assigned to this command, you can assign your own custom command using the **Keyboard** page of the **Options** dialog box. For more information, see [Identify and customize keyboard shortcuts](#).

To switch to specific files in the editor

- Press **Ctrl+Tab** to display the **IDE Navigator**. Hold down the **Ctrl** key and press **Tab** repeatedly until you select the file you intend to switch to.

TIP

To reverse the order in which you go through the **Active Files** list, hold down the **Ctrl+Shift** keys and press **Tab**.

- or -

- In the upper right corner of the editor, choose the **Active Files** button, and then select a file from the list to switch to.

- or -

- On the menu bar, choose **Window > Windows**.
- In the list, select the file you want to view and then choose **Activate**.

Navigate among tool windows in the IDE

The **IDE Navigator** also lets you cycle through the tool windows you have open in the IDE. You can use either of two commands to access the **IDE Navigator** to cycle through tool windows, depending on the order in which you want to cycle through. `Window.PreviousToolWindowNav` allows you to move to the file most recently accessed and `Window.NextToolWindowNav` allows you to move in the reverse order. **General Development Settings** assigns **Shift+Alt+F7** to `Window.PreviousDocumentWindowNav` and **Alt+F7** to `Window.NextDocumentWindowNav`.

NOTE

If the settings combination you are using does not already have a shortcut key combination assigned to this command, you can assign your own custom command using the **Keyboard** page of the **Options** dialog box. For more information, see [Identify and customize keyboard shortcuts](#).

To switch to a specific tool window in the IDE

- Press **Alt+F7** to display the **IDE Navigator**. Hold down the **Alt** key and press **F7** repeatedly until you select the window you intend to switch to.

TIP

To reverse the order in which you go through the **Active Tool Windows** list, hold down the **Shift+Alt** keys and press **F7**.

See also

- [Customize window layouts](#)
- [Default keyboard shortcuts](#)

Solutions and projects in Visual Studio

10/18/2019 • 3 minutes to read • [Edit Online](#)

This page describes the concept of a *project* and a *solution* in Visual Studio. It also briefly covers the Solution Explorer tool window and how to create a new project.

NOTE

This topic applies to Visual Studio on Windows. For Visual Studio for Mac, see [Projects and solutions in Visual Studio for Mac](#).

Projects

When you create an app or website in Visual Studio, you start with a *project*. In a logical sense, a project contains all files that are compiled into an executable, library, or website. Those files can include source code, icons, images, data files, and so on. A project also contains compiler settings and other configuration files that might be needed by various services or components that your program communicates with.

Project file

Visual Studio uses [MSBuild](#) to build each project in a solution, and each project contains an MSBuild project file. The file extension reflects the type of project, for example, a C# project (.csproj), a Visual Basic project (.vbproj), or a database project (.dbproj). The project file is an XML document that contains all the information and instructions that MSBuild needs in order to build your project, including the content, platform requirements, versioning information, web server or database server settings, and the tasks to perform.

Project files are based on the [MSBuild XML schema](#). To look at the contents of newer, [sdk-style project files](#) in Visual Studio, right-click on the project node in **Solution Explorer** and select **Edit <projectname>**. To look at the contents of .NET Framework and other projects of that style, first unload the project (right-click on the project node in **Solution Explorer** and select **Unload Project**). Then, right-click on the project and choose **Edit <projectname>**.

NOTE

You don't have to use solutions or projects in Visual Studio to edit, build, and debug code. You can simply open the folder that contains your source files in Visual Studio and start editing. For more information, see [Develop code in Visual Studio without projects or solutions](#).

Solutions

A project is contained within a *solution*. Despite its name, a solution is not an "answer". It's simply a container for one or more related projects, along with build information, Visual Studio window settings, and any miscellaneous files that aren't associated with a particular project. A solution is described by a text file (extension .sln) with its own unique format; it's not intended to be edited by hand.

Visual Studio uses two file types (.sln and .suo) to store settings for solutions:

EXTENSION	NAME	DESCRIPTION
-----------	------	-------------

Extension	Name	Description
.sln	Visual Studio Solution	Organizes projects, project items, and solution items in the solution.
.suo	Solution User Options	Stores user-level settings and customizations, such as breakpoints.

Create new projects

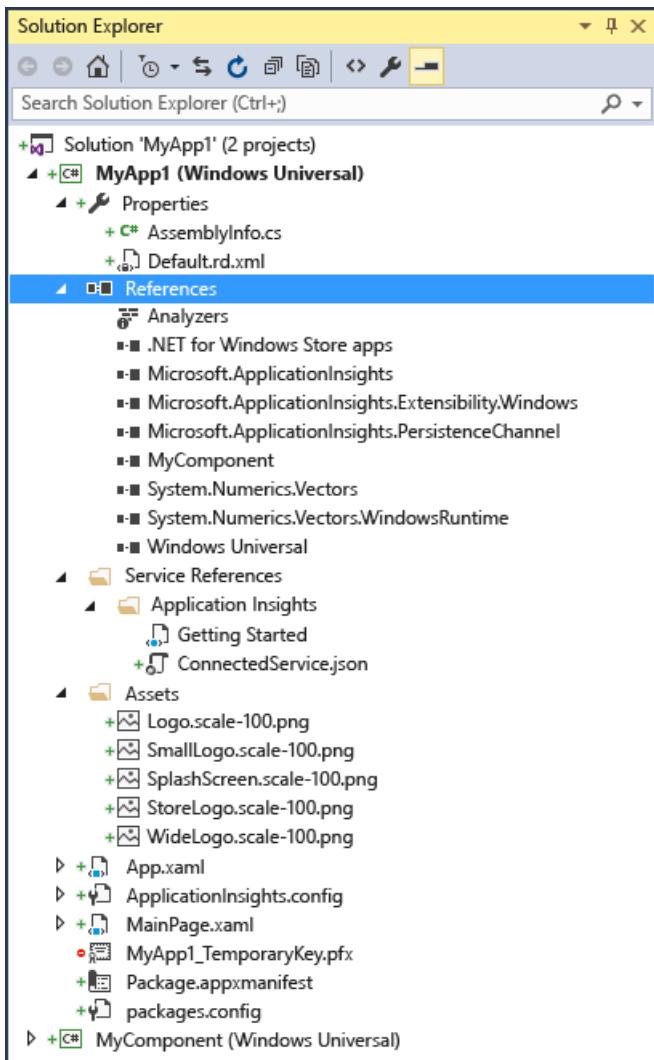
The easiest way to create a new project is to start from a project template for a particular type of application or website. A project template consists of a basic set of pre-generated code files, config files, assets, and settings. These templates are available in the dialog box where you create a new project (**File > New > Project**). For more information, see [Create a new project in Visual Studio](#) and [Create solutions and projects](#).

If you often customize your projects in a certain way, you can create a custom project template that you can then use to create new projects from. For more information, see [Create project and item templates](#).

When you create a new project, it is saved by default at `%USERPROFILE%\source\repos`. You can change this location in the **Projects location** setting under **Tools > Options > Projects and Solutions > Locations**. For more information, see [Projects and Solutions page, Options dialog box](#).

Solution Explorer

After you create a new project, you can use **Solution Explorer** to view and manage the project and solution and their associated items. The following illustration shows **Solution Explorer** with a C# solution that contains two projects:



Many menu commands are available from the right-click menu on various items in **Solution Explorer**. These commands include building a project, managing NuGet packages, adding a reference, renaming a file, and running tests, just to name a few. The toolbar across the top of **Solution Explorer** has buttons to switch from a solution view to a folder view, show hidden files, collapse all nodes, and more.

For ASP.NET Core projects, you can customize how files are nested in **Solution Explorer**. For more information, see [Customize file nesting in Solution Explorer](#).

See also

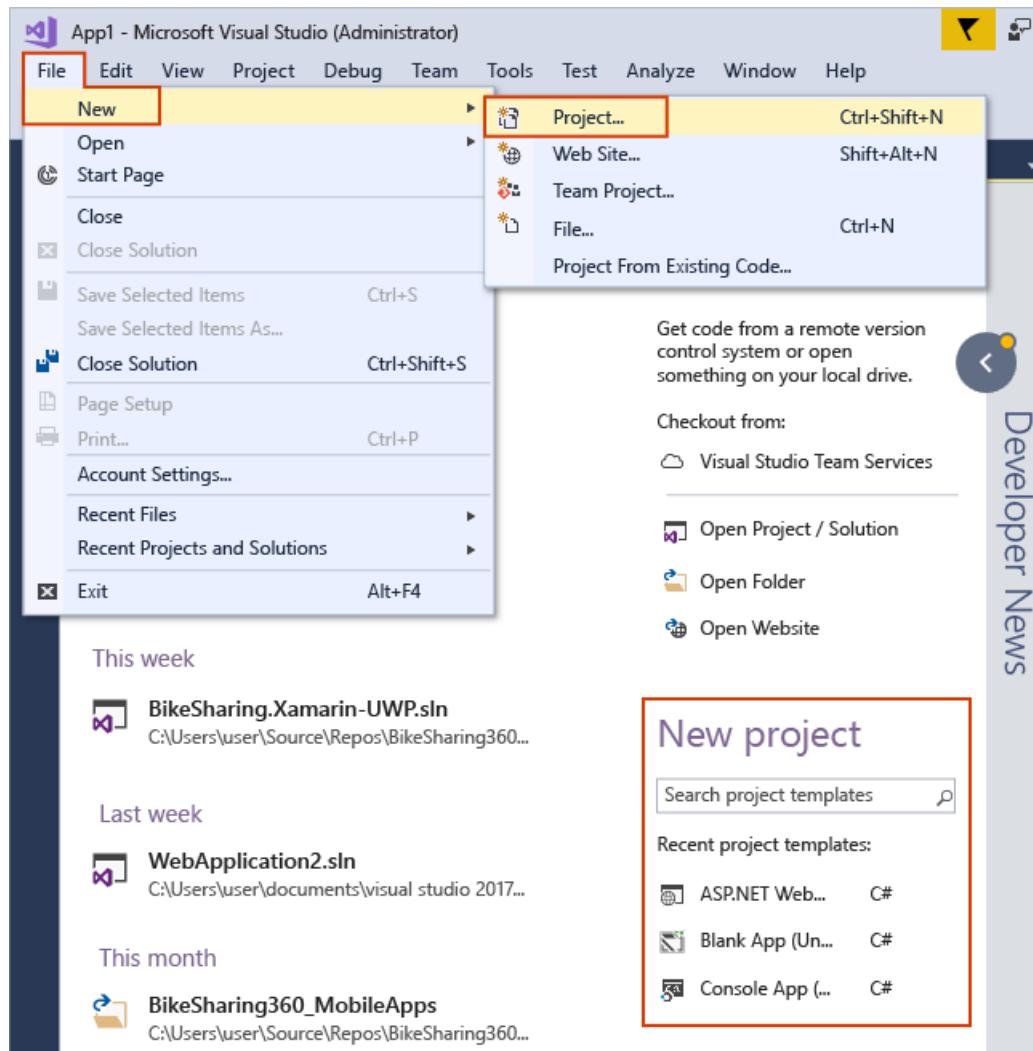
- [Visual Studio IDE](#)
- [Projects and solutions \(Visual Studio for Mac\)](#)
- [Add and remove project items \(Visual Studio for Mac\)](#)

Create a new project in Visual Studio

10/18/2019 • 3 minutes to read • [Edit Online](#)

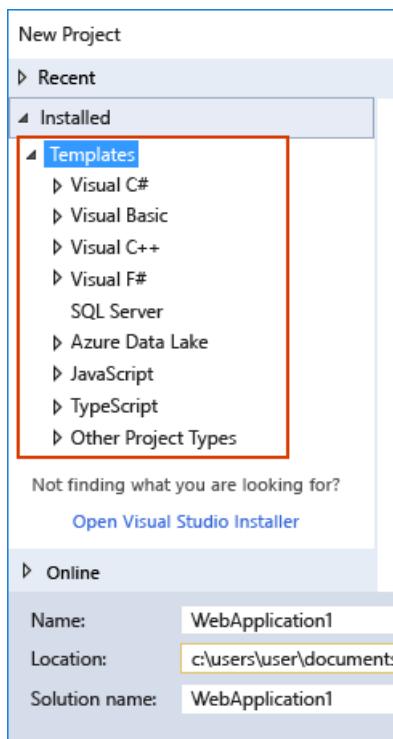
Open the New Project dialog

There are multiple ways to create a new project in Visual Studio 2017. On the Start page, you can type in the name of a project template in the **Search project templates** box or choose the **Create new project** link to open the **New Project** dialog box. Aside from the Start page, you can also choose **File > New > Project** on the menu bar or click the **New Project** button on the toolbar.



Select a template type

In the **New Project** dialog box, available project templates appear in a list under the **Templates** category. Templates are organized by programming language and project type, such as Visual C#, JavaScript, and Azure Data Lake.

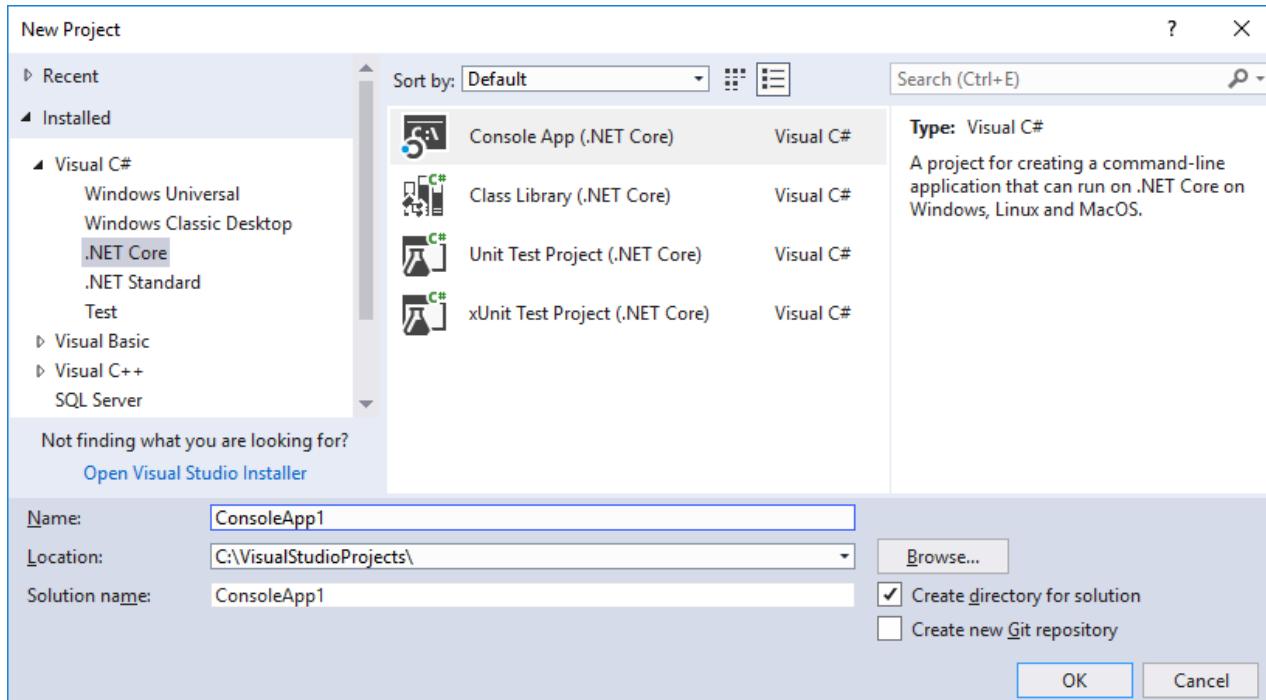


NOTE

The list of available languages and project templates that appears depends on the version of Visual Studio you are running and the workloads that are installed. To learn about how to install additional workloads, see [Modify Visual Studio by adding or removing workloads and components](#).

Show the list of templates for the programming language you want to use by clicking the triangle next to the language name and then choosing a project category (such as Windows Desktop).

The following image shows the project templates available for Visual C# .NET Core projects:



Configure your project

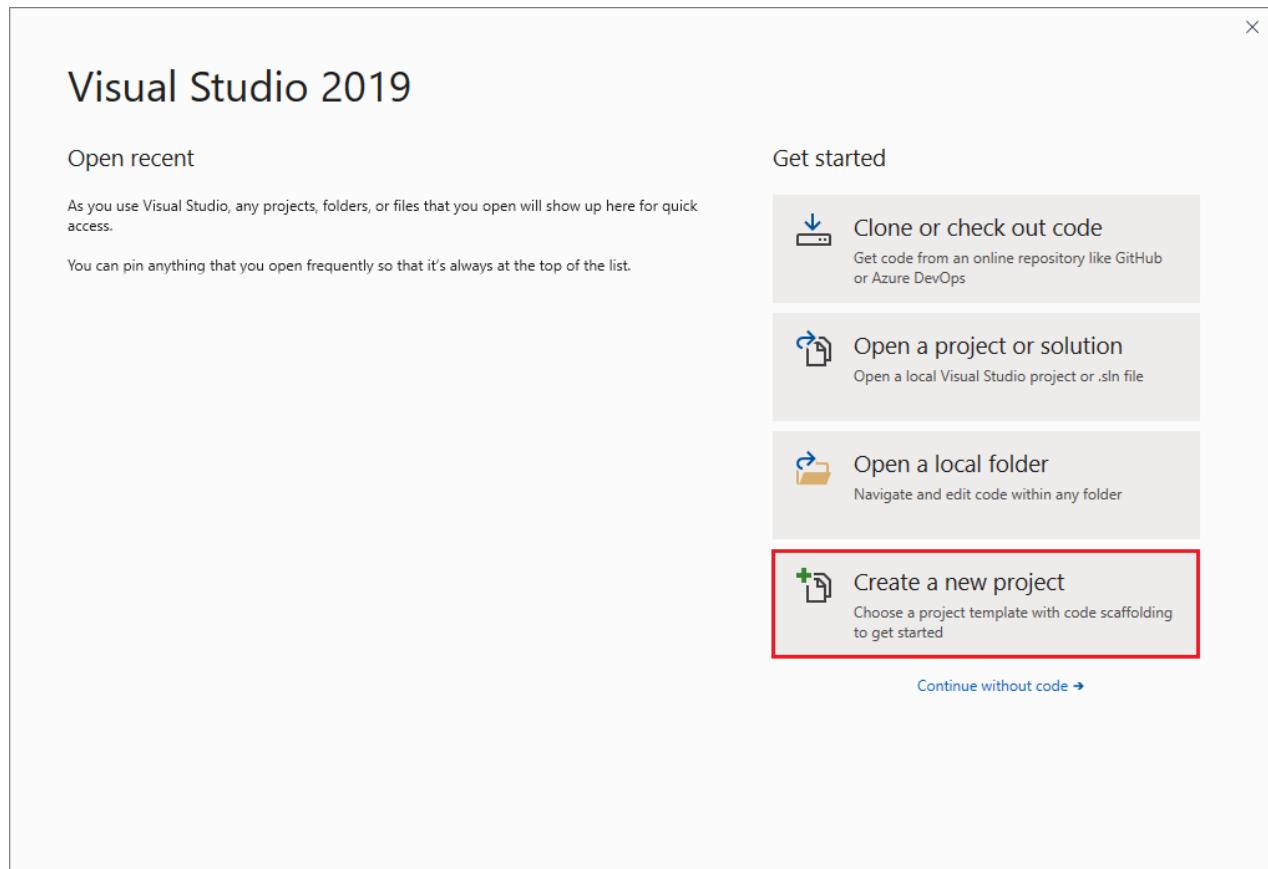
Enter a name for the new project in the **Name** box. You can save the project in the default location on your computer or click the **Browse** button to find another location. You can also choose a solution name or add the new

project to a Git repository (by choosing **Add to Source Control**).

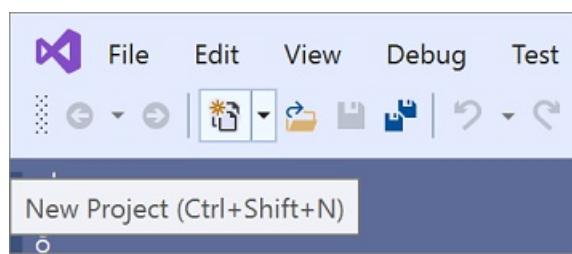
Click **OK** to create the solution and project.

Open the Create a new project page

There are multiple ways to create a new project in Visual Studio 2019. When you first open Visual Studio, the start window appears, and from there, you can choose **Create a new project**.



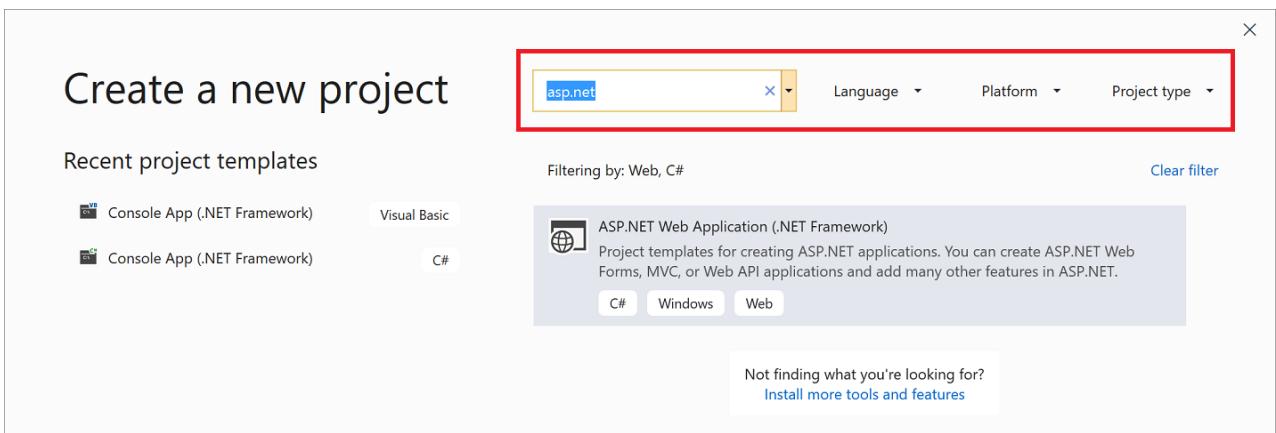
If the Visual Studio development environment is already open, you can create a new project by choosing **File > New > Project** on the menu bar or by clicking the **New Project** button on the toolbar.



Select a template type

On the **Create a new project** page, a list of your recently selected templates appears on the left. The templates are sorted by *most recently used*.

If you're not selecting from the recently used templates, you can filter all available project templates by **Language** (for example, C# or C++), **Platform** (for example, Windows or Azure), and **Project type** (for example, Desktop or Web). You can also enter search text into the search box to further filter the templates, for example, **asp.net**.



The tags that appear under each template correspond to the three dropdown filters (Language, Platform, and Project type).

TIP

If you don't see the template you're looking for, you may be missing a workload for Visual Studio. To install additional workloads, for example, **Azure Development** or **Mobile Development with .NET**, click the **Install more tools and features** link to open Visual Studio Installer. From there, select the workloads you want to install, and then choose **Modify**. After that, additional project templates will be available to choose from.

Not finding what you're looking for?
[Install more tools and features](#)

Select a template and then click **Next**.

Configure your project

The **Configure your new project** page has options to name your project (and solution), choose a disk location, and select a Framework version (if applicable to the template you chose).



Configure your new project

ASP.NET Web Application (.NET Framework) C# Windows Web

Project name

my-web-site

Location

C:\VSProjects\



Solution name ⓘ

web-solution

Place solution and project in the same directory

Framework

.NET Framework 4.7.2

Back

Create

NOTE

If you create a new project when you already have a project or solution open in Visual Studio, an extra configuration option is available. You can choose to create a new solution or add the new project to the solution that's already open.

Solution

Create new solution

Create new solution

Add to solution

Click **Create** to create the new project.

Add additional projects to a solution

If you want to add an additional project to a solution, right-click the solution node in **Solution Explorer** and choose **Add > New Project**.

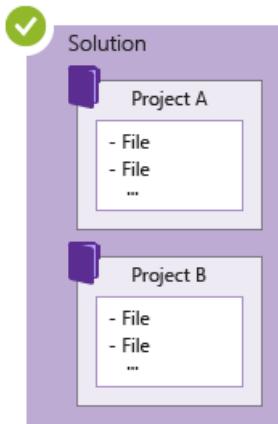
See also

- [Create solutions and projects](#)

Create solutions and projects

10/21/2019 • 5 minutes to read • [Edit Online](#)

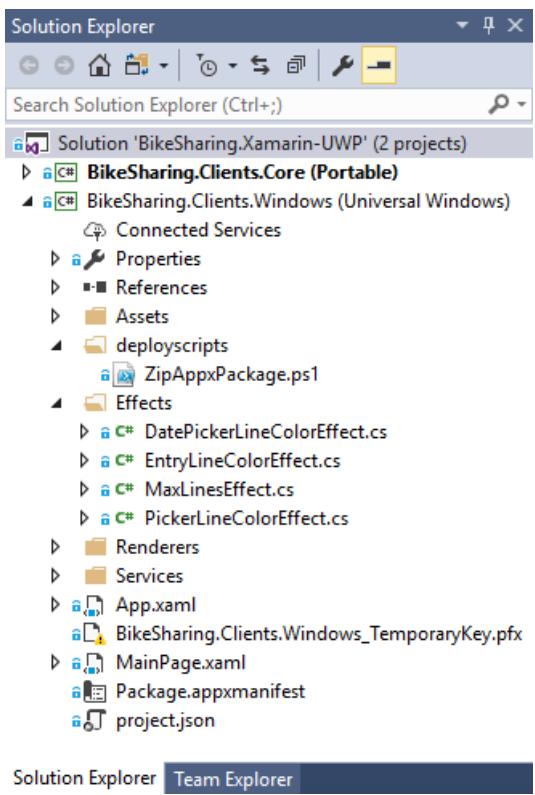
Projects hold the items needed to build your app in Visual Studio, such as source code files, bitmaps, icons, and component and service references. When you create a new project, Visual Studio creates a *solution* to contain the project. You can then add other new or existing projects to the solution if you want. Solutions can also contain files that aren't connected to any specific project.



NOTE

This topic applies to Visual Studio on Windows. For Visual Studio for Mac, see [Create projects in Visual Studio for Mac](#).

You can view your solutions and projects in a tool window called **Solution Explorer**. The following screenshot shows an example solution in **Solution Explorer (BikeSharing.Xamarin-UWP)** that contains two projects: **BikeSharing.Clients.Core** and **BikeSharing.Clients.Windows**. Each project contains multiple files, folders, and references. The project name in bold is the *startup project*; that is, the project that starts when you run the app. You can specify which project is the startup project.



Solution Explorer Team Explorer

While you can construct a project yourself by adding the necessary files to it, Visual Studio offers a selection of project templates to give you a head start. Creating a new project from a template gives you a project with the essentials for that project type, and you can rename the files or add new or existing code and other resources to it as needed.

That being said, solutions and projects are not required to develop apps in Visual Studio. You can also just open code that you have cloned from Git or downloaded elsewhere. For more information, see [Develop code in Visual Studio without projects or solutions](#).

Create a project from a project template

For information about creating a new project from a template, see [Create a new project in Visual Studio](#).

Create a project from existing code files

If you have a collection of code source files, you can easily add them to a project.

1. On the menu, choose **File > New > Project From Existing Code**.
2. In the **Create Project from Existing Code Files** wizard, choose the project type you want in the **What type of project would you like to create?** drop-down list box, and then choose the **Next** button.
3. In the wizard, browse to the location of the files and then enter a name for the new project in the **Name** box. When you are done, choose the **Finish** button.

NOTE

This option works best for a relatively simple collection of files. Currently, only C++, Apache Cordova, Visual Basic, and C# project types are supported.

Add files to a solution

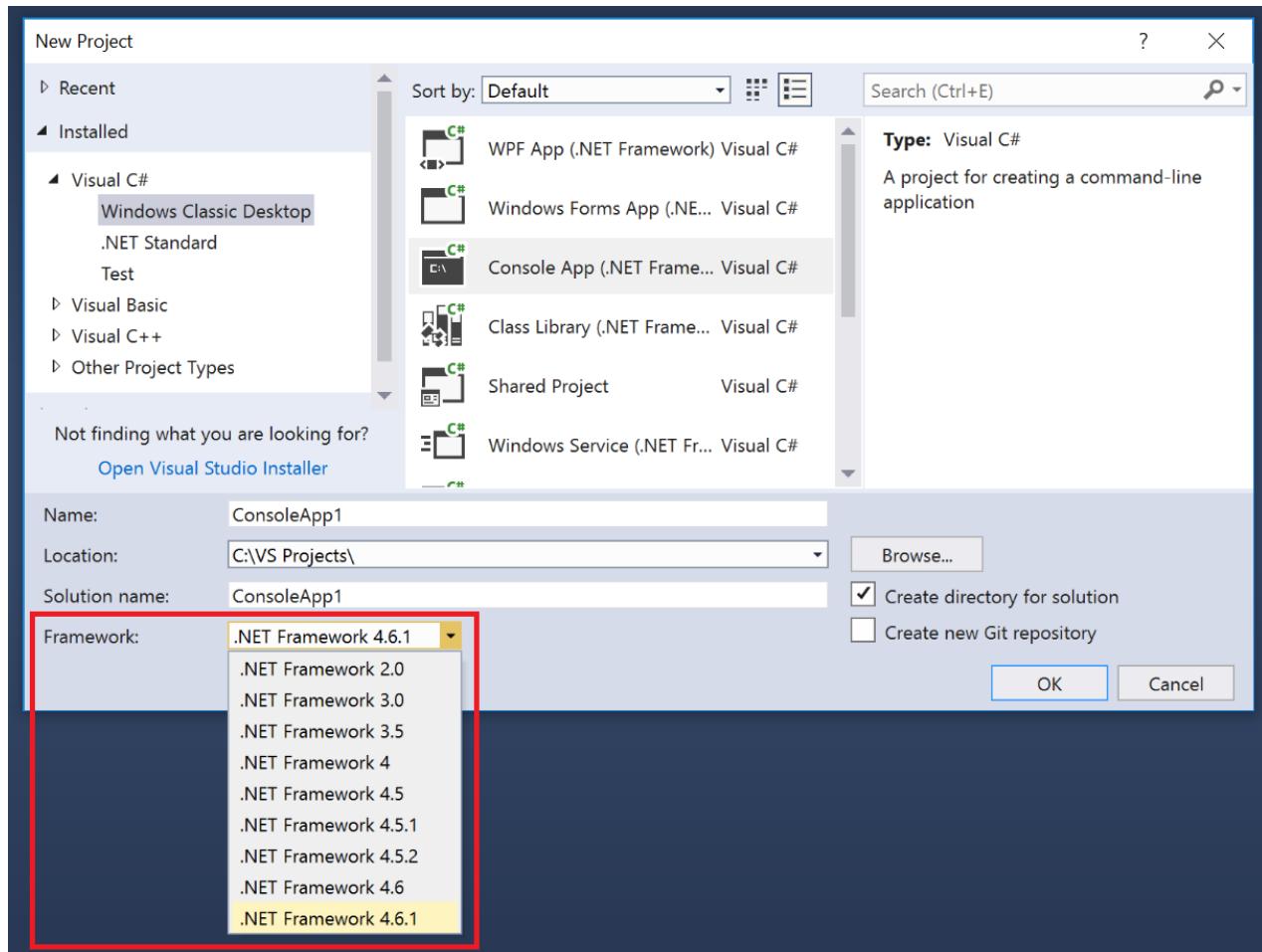
If you have a file that applies to multiple projects, such as a readme file for the solution, or other files that logically

belong at the solution level rather than under a specific project, then you can add them to the solution itself. To add an item to a solution, on the context (right-click) menu of the solution node in **Solution Explorer**, choose **Add > New Item**, or **Add > Existing Item**.

Create a .NET project that targets a specific version of the .NET Framework

When you create a .NET Framework project, you can specify a specific version of the .NET Framework that you want the project to use. (When you create a .NET Core project, you don't specify a framework version.)

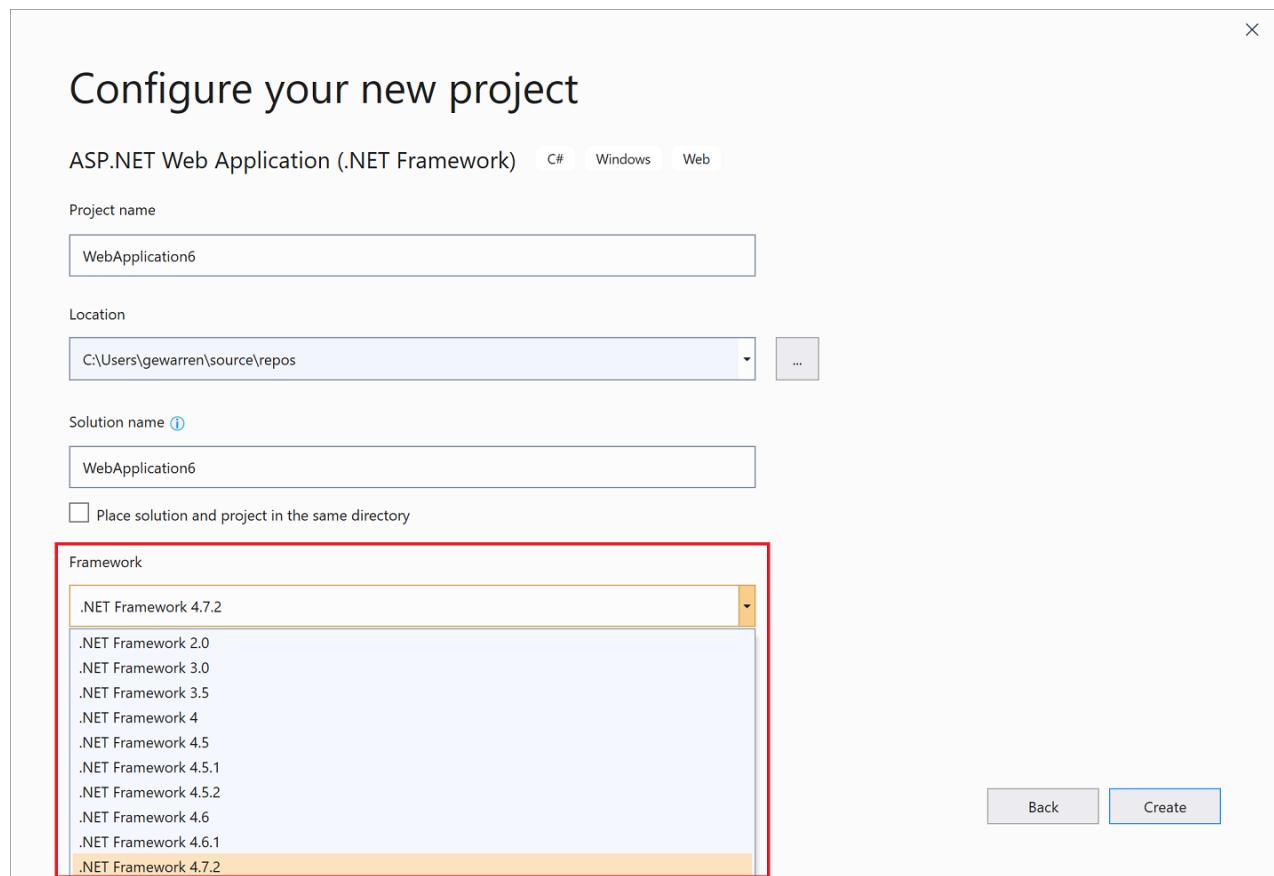
To specify a .NET Framework version, choose the **Framework** drop-down menu in the **New Project** dialog box.



NOTE

You must have .NET Framework 3.5 installed on your system to access .NET Framework versions earlier than .NET Framework 4.

To specify a .NET Framework version, choose the **Framework** drop-down menu on the **Create a new project** page.



Create empty solutions

You can also create empty solutions that have no projects. This might be preferable in cases where you want to construct your solution and projects from scratch.

To create an empty solution

1. On the menu bar, choose **File > New > Project**.
2. In the left (**Templates**) pane, choose **Other Project Types > Visual Studio Solutions** in the expanded list.
3. In the middle pane, choose **Blank Solution**.
4. Enter **Name** and **Location** values for your solution, and then choose **OK**.
2. On the **Create a new project** page, type **solution** into the search box.
3. Select the **Blank Solution** template, and then click **Next**.
4. Enter **Name** and **Location** values for your solution, and then choose **Create**.

After you create an empty solution, you can add new or existing projects or items to it by choosing **Add New Item** or **Add Existing Item** on the **Project** menu.

As mentioned earlier, you can also open code files without needing a project or solution. To learn about developing code in this way, see [Develop code in Visual Studio without projects or solutions](#).

Create a temporary project

(C# and Visual Basic only)

If you create a .NET-based project without specifying a disk location, it is a temporary project. Temporary projects enable you to experiment with .NET projects. At any time while you are working with a temporary project, you can

choose to save it or discard it.

To create a temporary project, first go to **Tools > Options > Projects and Solutions > General**, and uncheck the **Save new projects when created** checkbox. Then open the **New Project** dialog box as usual.

Delete a solution, project, or item

You can delete solutions and their contents permanently, but not by using the Visual Studio IDE. Deleting items within Visual Studio only removes them from the current solution or project. To permanently delete a solution or other component from your system, use File Explorer to delete the folder that contains the `.sln` and `.suo` solution files. However, before permanently deleting a solution, it's recommended that you back up any projects or files in case you need them again.

NOTE

The `.suo` file is a hidden file that is not displayed under the default File Explorer settings. To show hidden files, on the **View** menu in File Explorer, select the **Hidden Items** checkbox.

Permanently delete a solution

1. In **Solution Explorer**, on the right-click menu (context menu) of the solution you want to delete, choose **Open folder in File Explorer**.
2. In File Explorer, navigate up one level.
3. Choose the folder containing the solution and then press the **Delete** key.

See also

- [Solutions and projects](#)
- [Microsoft's open source repositories on GitHub](#)
- [Developer code samples](#)
- [Create projects \(Visual Studio for Mac\)](#)

Tutorial: Open a project from a repo

8/30/2019 • 3 minutes to read • [Edit Online](#)

In this tutorial, you'll use Visual Studio to connect to a repository for the first time and then open a project from it.

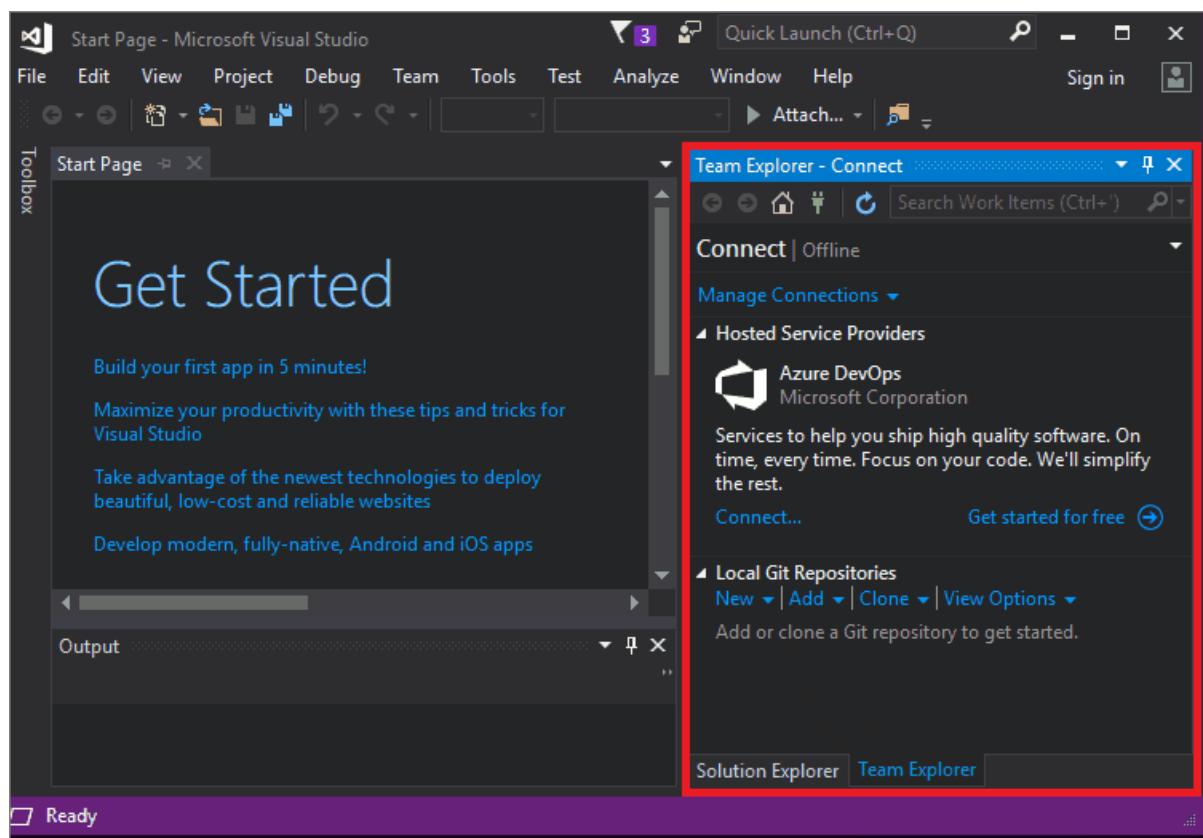
If you haven't already installed Visual Studio, go to the [Visual Studio downloads](#) page to install it for free.

If you haven't already installed Visual Studio, go to the [Visual Studio downloads](#) page to install it for free.

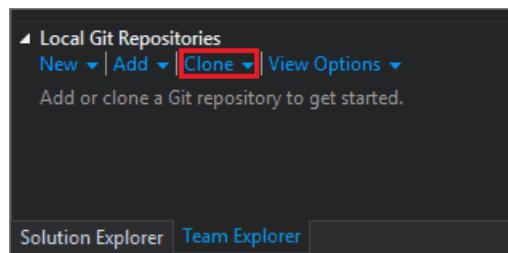
Open a project from a GitHub repo

1. Open Visual Studio 2017.
2. From the top menu bar, choose **File > Open > Open from Source Control**.

The **Team Explorer - Connect** pane opens.



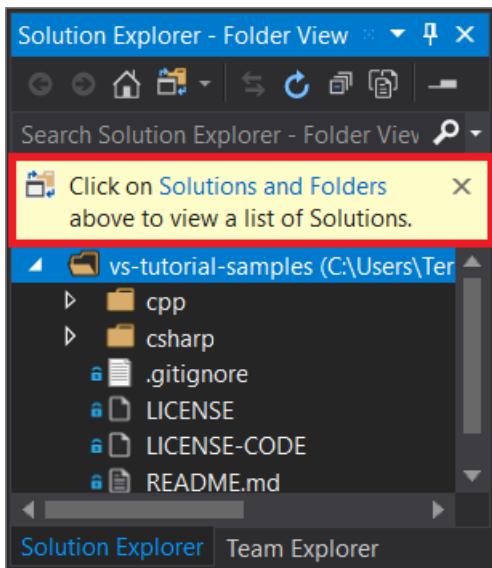
3. In the **Local Git Repositories** section, choose **Clone**.



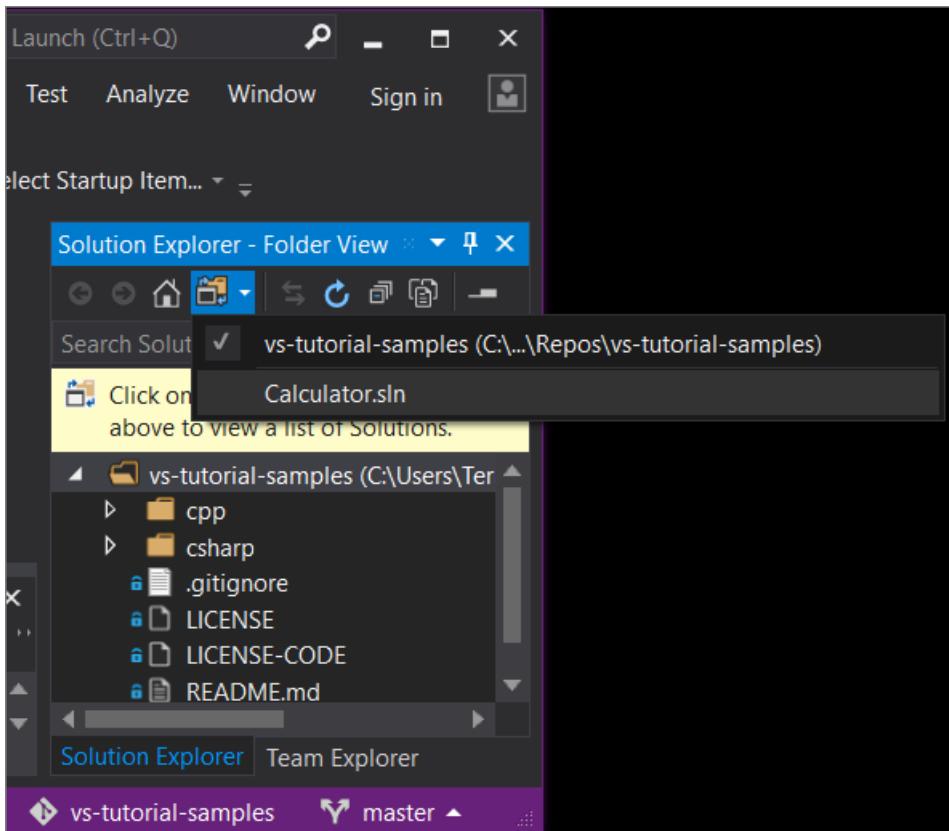
4. In the box that says **Enter the URL of a Git repo to clone**, type or paste the URL for your repo, and then press **Enter**. (You might receive a prompt to sign in to GitHub; if so, do so.)

After Visual Studio clones your repo, Team Explorer closes and Solution Explorer opens. A message appears

that says *Click on Solutions and Folders above to view a list of Solutions*. Choose **Solutions and Folders**.



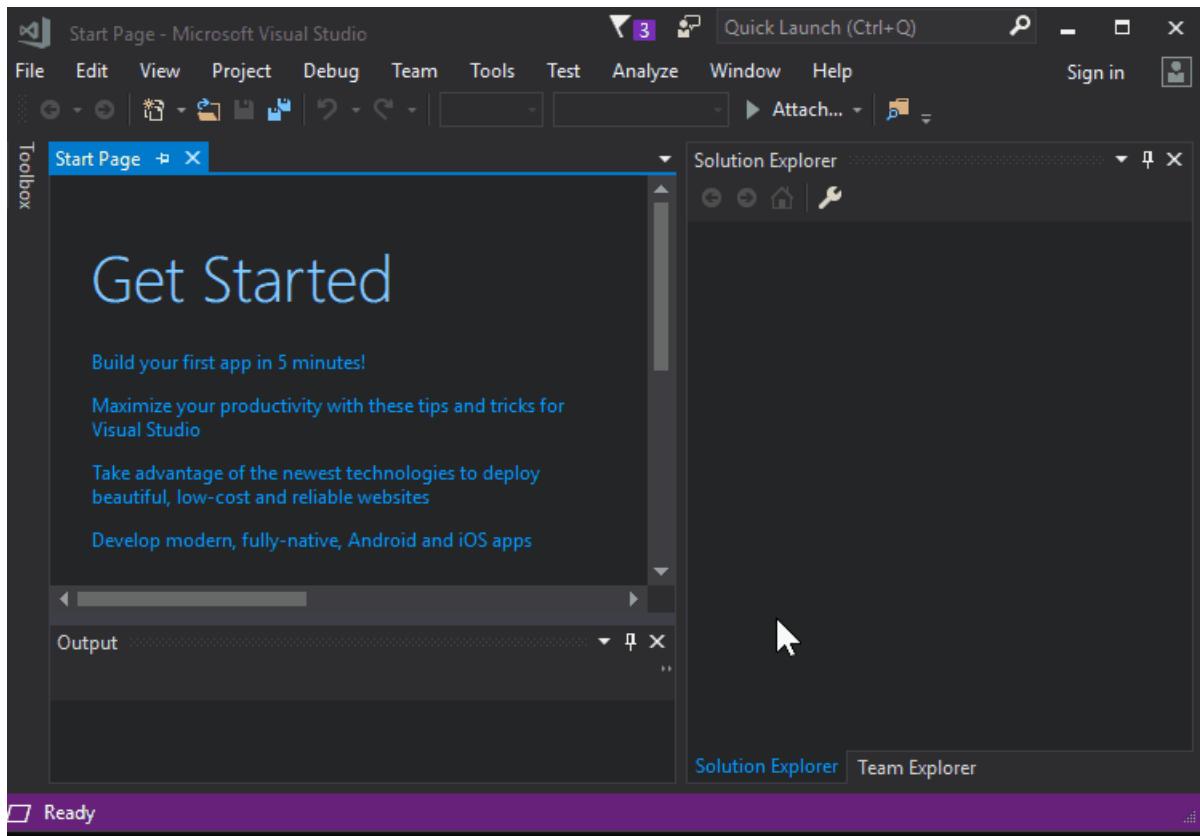
5. If you have a solution file available, it will appear in the "Solutions and Folders" fly-out menu. Choose it, and Visual Studio opens your solution.



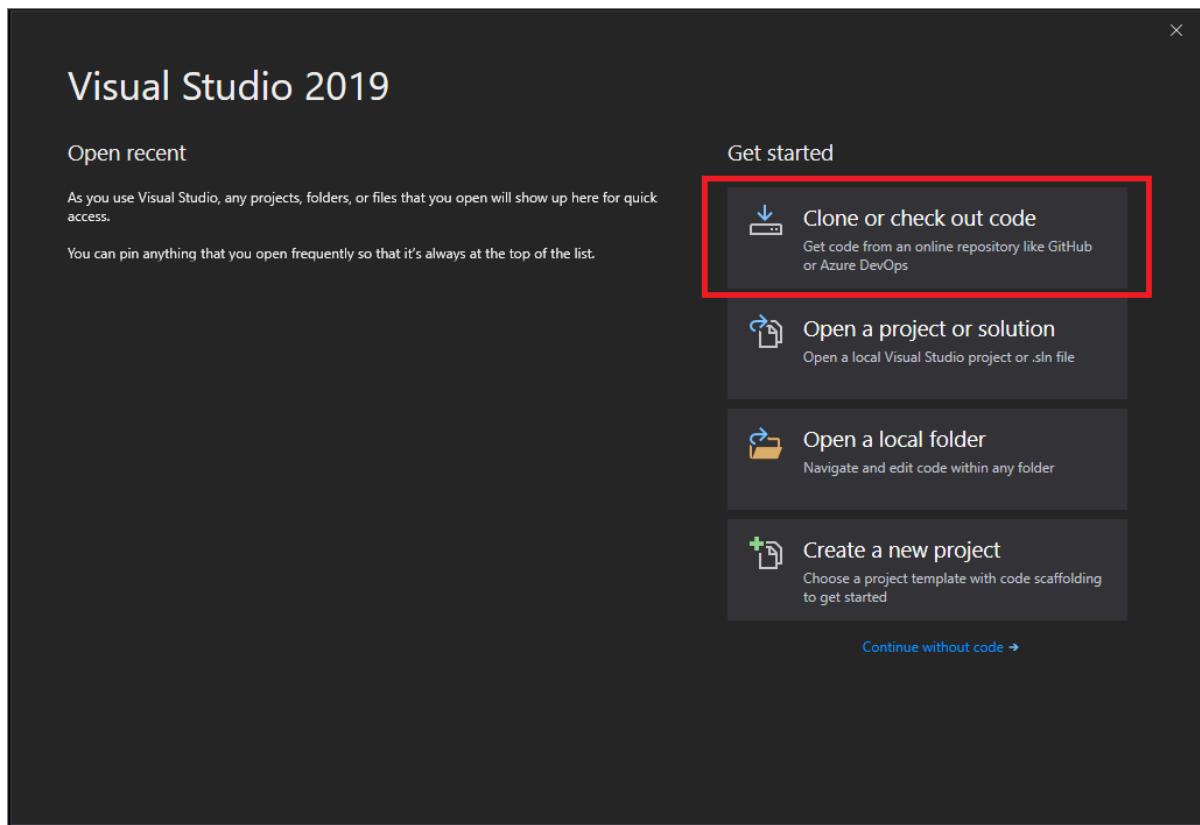
If you do not have a solution file (specifically, a .sln file) in your repo, the fly-out menu will say "No Solutions Found." However, you can double-click any file from the folder menu to open it in the Visual Studio code editor.

Review your work

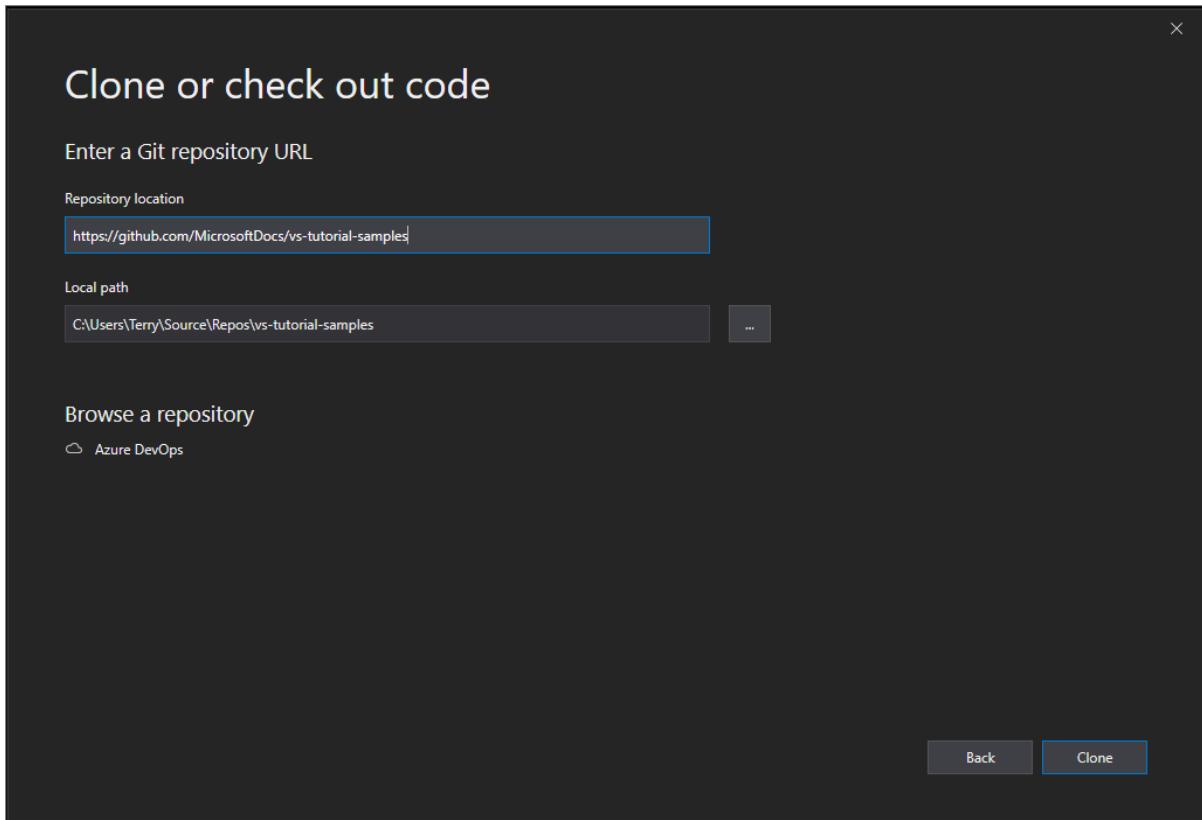
View the following animation to check the work that you completed in the previous section.



1. Open Visual Studio 2019.
2. On the start window, choose **Clone or check out code**.

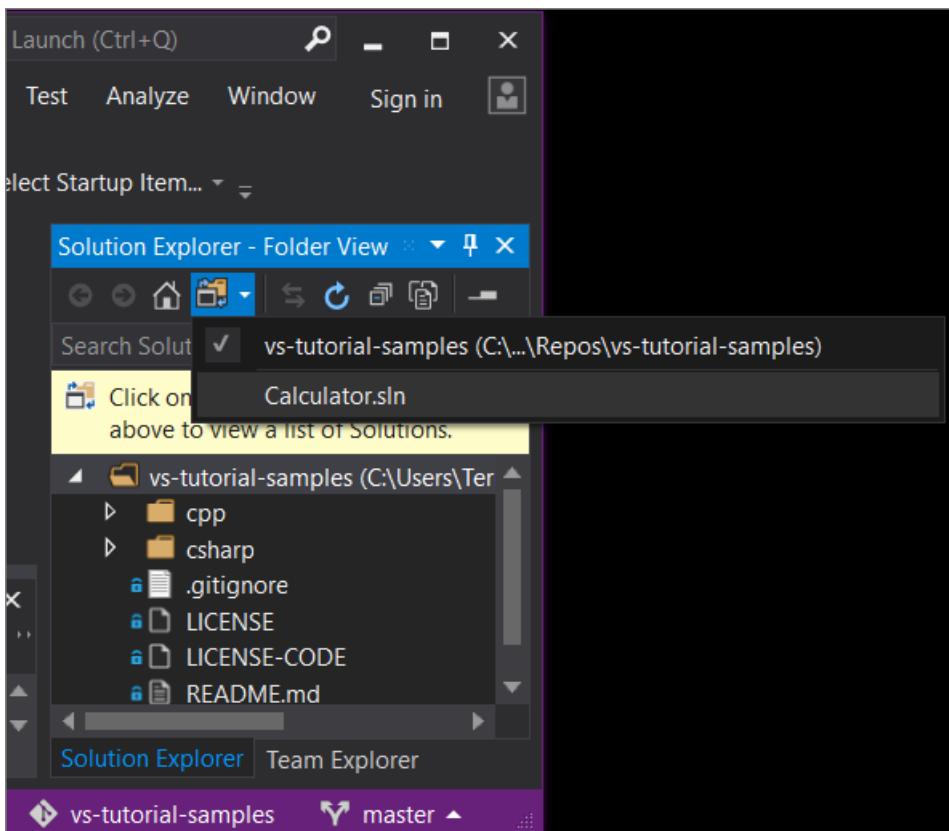


3. Enter or type the repository location, and then choose **Clone**.



Visual Studio opens the project from the repo.

4. If you have a solution file available, it will appear in the "Solutions and Folders" fly-out menu. Choose it, and Visual Studio opens your solution.

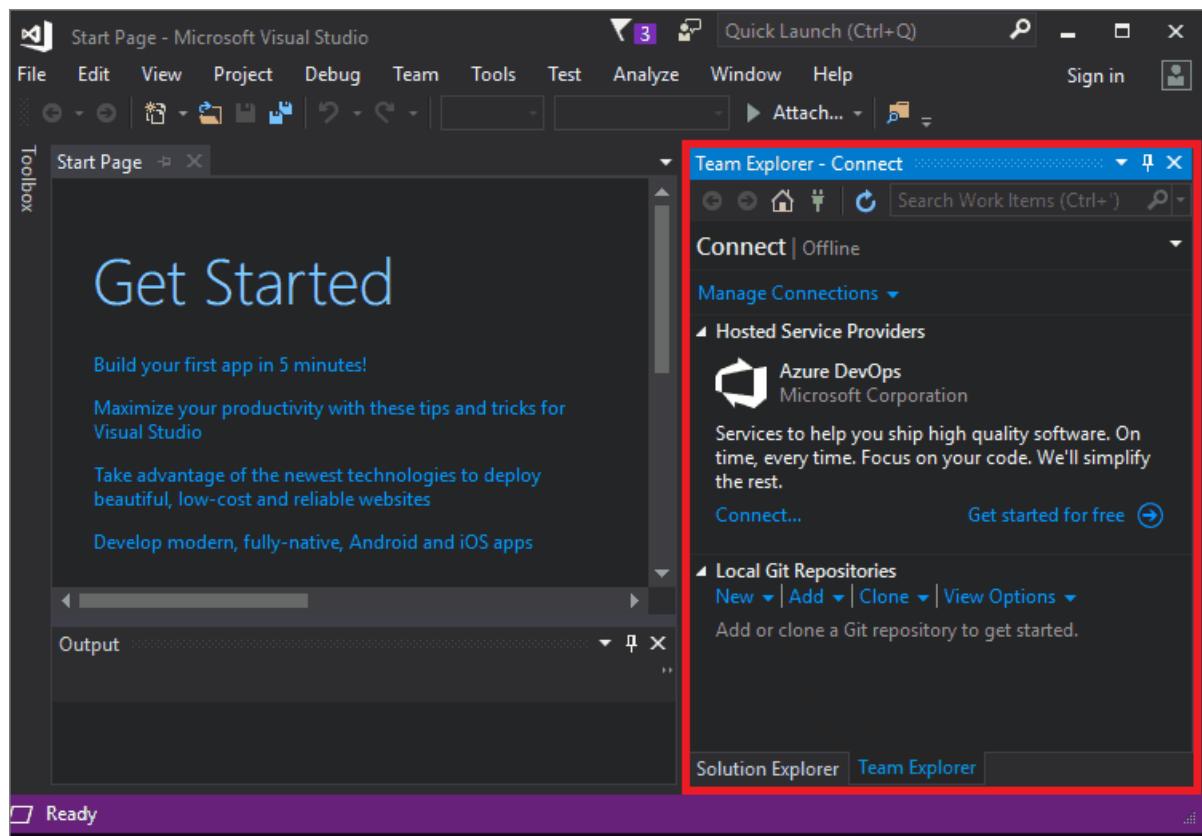


If you do not have a solution file (specifically, a .sln file) in your repo, the fly-out menu will say "No Solutions Found." However, you can double-click any file from the folder menu to open it in the Visual Studio code editor.

Open a project from an Azure DevOps repo

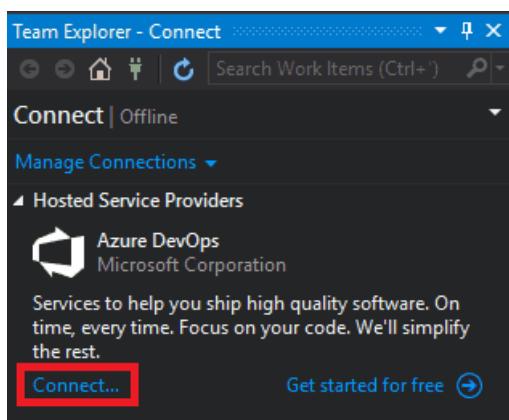
1. Open Visual Studio 2017.
2. From the top menu bar, choose **File > Open > Open from Source Control**.

The **Team Explorer - Connect** pane opens.

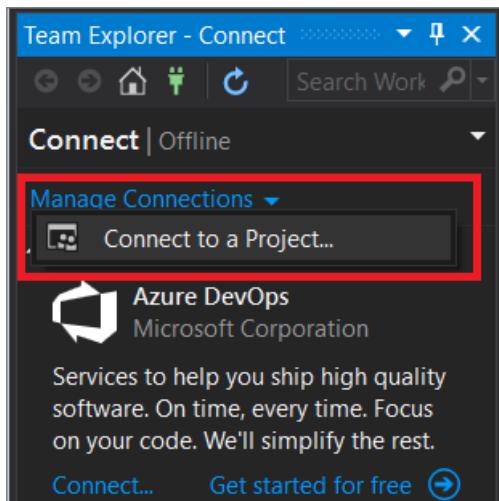


3. Here are two ways to connect to your Azure DevOps repo:

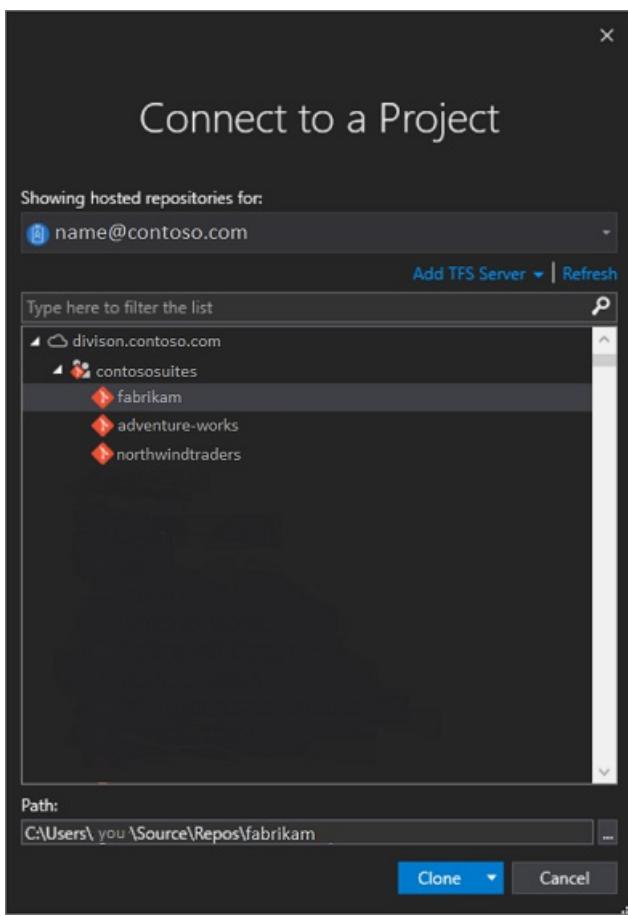
- In the **Hosted Service Providers** section, choose **Connect....**



- In the **Manage Connections** drop-down list, choose **Connect to a Project....**



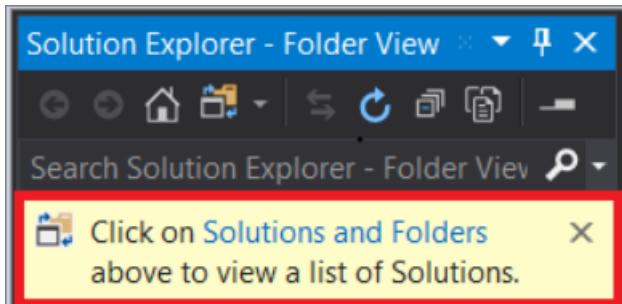
4. In the **Connect to a Project** dialog box, choose the repo that you want to connect to, and then choose **Clone**.



NOTE

What you see in the list box depends on the Azure DevOps repositories that you have access to.

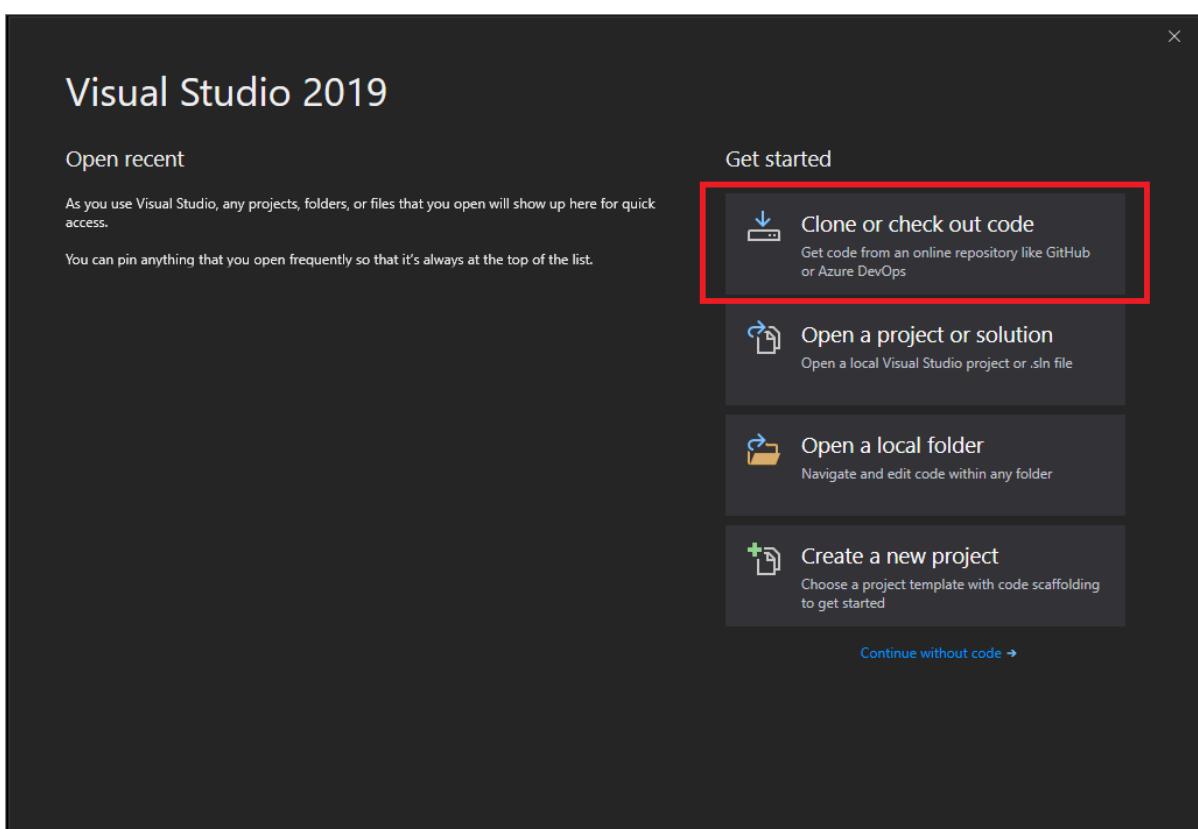
5. After Visual Studio clones your repo, Team Explorer closes and Solution Explorer opens. A message appears that says *Click on Solutions and Folders above to view a list of Solutions*. Choose **Solutions and Folders**.



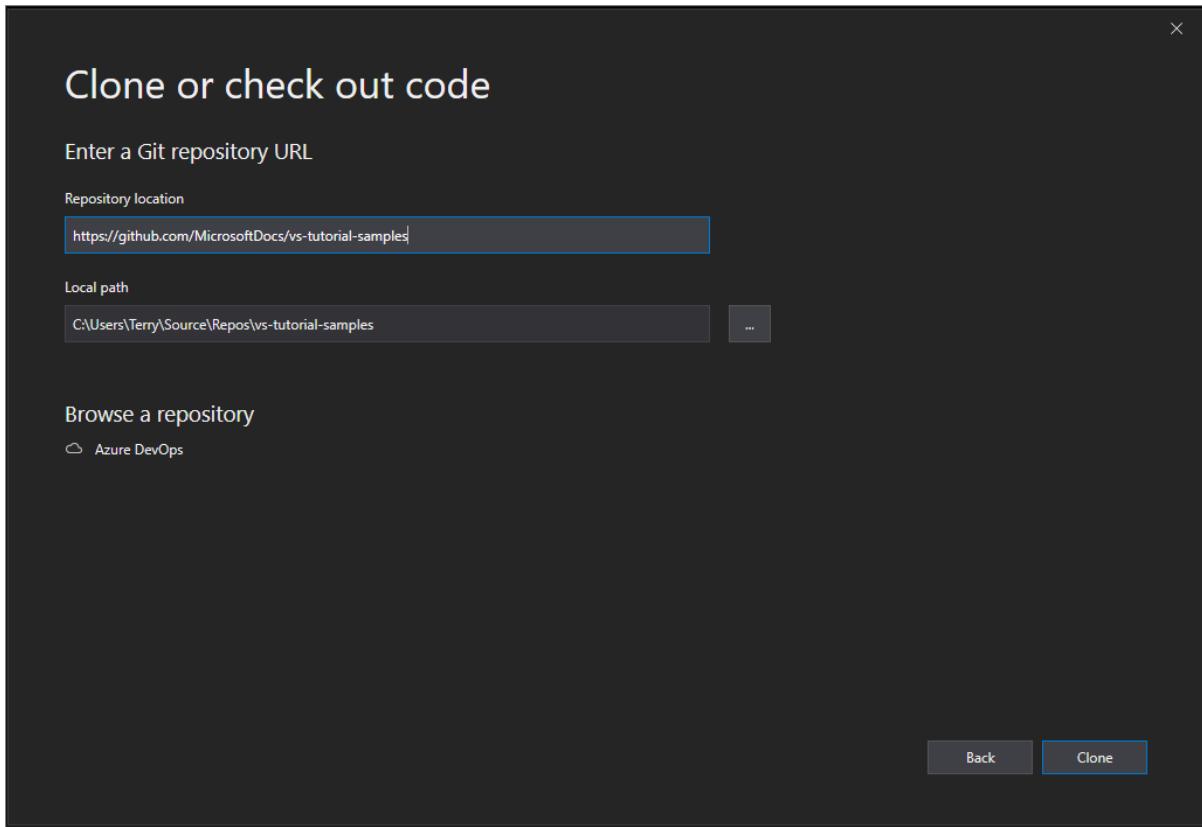
A solution file (specifically, a .sln file), will appear in the "Solutions and Folders" fly-out menu. Choose it, and Visual Studio opens your solution.

If you do not have a solution file in your repo, the fly-out menu will say "No Solutions Found". However, you can double-click any file from the folder menu to open it in the Visual Studio code editor.

1. Open Visual Studio 2019.
2. On the start window, choose **Clone or check out code**.

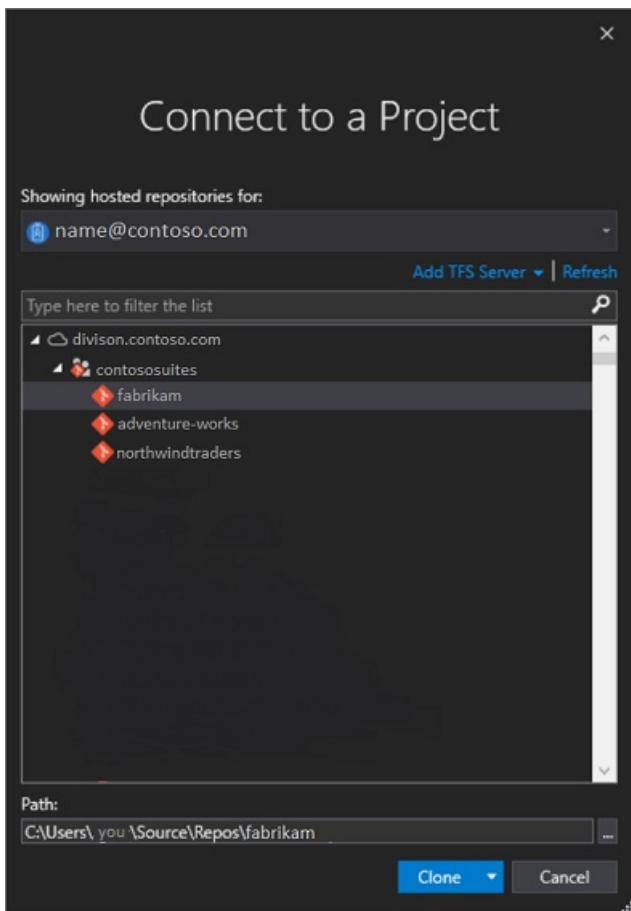


3. In the **Browse a repository** section, choose **Azure DevOps**.



If you see a sign-in window, sign in to your account.

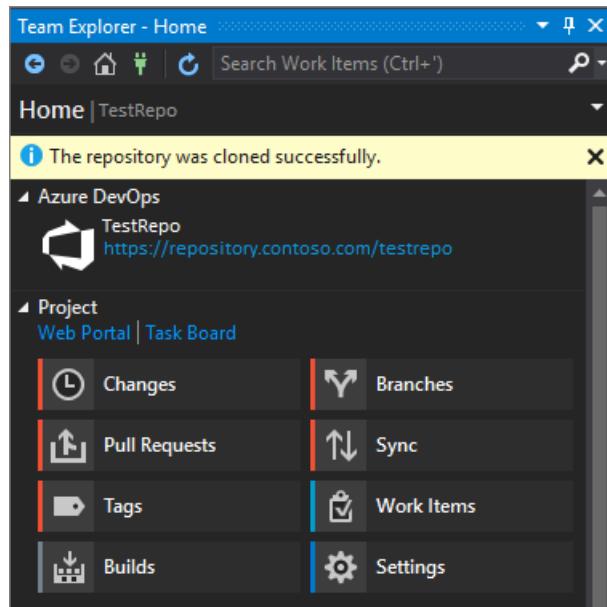
4. In the **Connect to a Project** dialog box, choose the repo that you want to connect to, and then choose **Clone**.



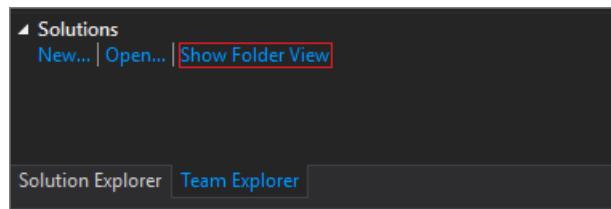
NOTE

What you see in the list box depends on the Azure DevOps repositories that you have access to.

Visual Studio opens **Team Explorer** and a notification appears when the clone is complete.

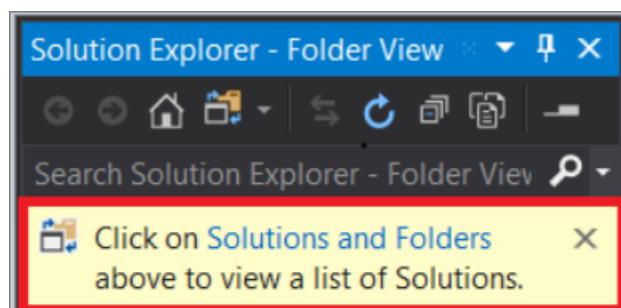


5. To view your folders and files, choose the **Show Folder View** link.



Visual Studio opens **Solution Explorer**.

6. Choose the **Solutions and Folders** link to search for a solution file (specifically, a .sln file) to open.



If you do not have a solution file in your repo, a "No Solutions Found" message appears. However, you can double-click any file from the folder menu to open it in the Visual Studio code editor.

Next steps

If you're ready to code with Visual Studio, dive into any of the following language-specific tutorials:

- [Visual Studio tutorials | C#](#)
- [Visual Studio tutorials | Visual Basic](#)
- [Visual Studio tutorials | C++](#)
- [Visual Studio tutorials | Python](#)

- Visual Studio tutorials | **JavaScript**, **TypeScript**, and **Node.js**

See also

- Azure DevOps Services: Get started with Azure Repos and Visual Studio
- Microsoft Learn: Get started with Azure DevOps

Project migration and upgrade reference for Visual Studio

10/24/2019 • 29 minutes to read • [Edit Online](#)

Each version of Visual Studio generally supports most previous types of projects, files, and other assets. You can work with them [as you always have](#), and provided that you don't depend on newer features, Visual Studio tries to preserve backwards compatibility with previous versions like Visual Studio 2015, Visual Studio 2013, and Visual Studio 2012. (See the [Release Notes](#) for which features are specific to which versions.)

Support for some project types also changes over time. A newer version of Visual Studio may no longer support certain projects at all, or requires updating a project such that it's no longer backwards compatible. For current status on migration issues, refer to the [Visual Studio Developer Community site](#).

This present article provides details only for project types that Visual Studio 2017 can migrate. The article excludes project types that are no longer supported in Visual Studio 2017 and cannot therefore be migrated. The article also excludes supported project types that have no migration issues; that list is found on [Platform Targeting and Compatibility](#).

IMPORTANT

Certain project types require installing the appropriate workloads through the Visual Studio installer. If you don't have the workload installed, Visual Studio reports an unknown or incompatible project type. In that case, check your installation options and try again. Again, see the [Platform Targeting and Compatibility](#) article for details on project support in Visual Studio 2017.

Project types

The following list describes support in Visual Studio 2017 for projects that were created in earlier versions.

If you don't see a project or file type listed here that should be, consult the [Visual Studio 2015 version of this article](#) and use the **Send feedback about > This page** button at the bottom of this page to provide details of your project. (If you use the anonymous "Is this page helpful?" control, we aren't able to respond to your feedback.)

TYPE OF PROJECT	SUPPORT
.NET Core projects (xproj)	Projects created with Visual Studio 2015 used preview tooling that included an xproj project file. In Visual Studio 2017, the xproj format is not supported other than for migration to csproj format. When you open an xproj file, you're prompted to migrate the file to the SDK-style csproj format. (A backup of the xproj file is made.) SDK-style csproj projects are not supported in Visual Studio 2015 and earlier. For more information, see Migrating .NET Core projects to the csproj format .

TYPE OF PROJECT	SUPPORT
ASP.NET Web Application and ASP.NET Core Web Application with Application Insights enabled	<p>For each Visual Studio user, resource information is stored in the registry per user instance. This information is used when a user doesn't have a project opened and wants to search Azure Application Insights data. Visual Studio 2015 uses different registry location than Visual Studio 2017 and does not conflict.</p> <p>Once a user creates an ASP.NET Web Application or ASP.NET Core Web Application, the resource is stored in the .suo file. The user can open the project in Visual Studio 2015 or 2017 and the resource information is used for both as long as Visual Studio supports projects and solutions being used across both versions. Users need to authenticate once on each product. For example, if a project is created with Visual Studio 2015 and opened in Visual Studio 2017, the user needs to authenticate on Visual Studio 2017.</p>
C#/Visual Basic Webform or Windows Form	You can open the project in Visual Studio 2017 and Visual Studio 2015.
Database Unit Test Projects (csproj, .vbproj)	<p>Older Data Unit test projects are loaded in Visual Studio 2017 but use the GAC'd version of dependencies. To upgrade the unit test project to use the latest dependencies, right-click on the project in Solution Explorer and select Convert to SQL Server Unit Testing Project...</p>
F#	<p>Visual Studio 2017 can open projects created in Visual Studio 2013 and 2015. To enable Visual Studio 2017 features in these projects, however, open the project properties and change target fsharp.core to F# 4.1. Note also that the F# language support option in the Visual Studio installer is not selected by default with .NET workloads; you must include it by selecting that option for the workload, or selecting it from the Individual components tab under Development activities.</p>
InstallShield MSI setup	<p>Installer projects created in Visual Studio 2010 can be opened in later versions with the help of the Visual Studio Installer Projects extension. Also see the WiX Toolset Visual Studio 2017 Extension. InstallShield Limited Edition is no longer included with Visual Studio. Check with Flexera Software about availability for Visual Studio 2017.</p>
LightSwitch	<p>LightSwitch is no longer supported in Visual Studio 2017. Projects created with Visual Studio 2012 and earlier opened in Visual Studio 2013 or Visual Studio 2015 are upgraded and can be opened only in Visual Studio 2013 or Visual Studio 2015 thereafter.</p>
Microsoft Azure Tools for Visual Studio	<p>To open these types of projects, first install the Azure SDK for .NET, then open the project. If necessary, your project is updated.</p>

TYPE OF PROJECT	SUPPORT
Model-View-Controller framework (ASP.NET MVC)	<p>Support for MVC versions and Visual Studio:</p> <ul style="list-style-type: none"> Visual Studio 2010 SP1 supports MVC 2 and MVC 3; MVC 4 support is added through the ASP.NET 4 MVC 4 for Visual Studio 2010 SP1 download Visual Studio 2012 supports only MVC 3 and MVC 4 Visual Studio 2013 supports only MVC 4 and MVC 5 Visual Studio 2017 and Visual Studio 2015 support MVC 4 (you can open existing projects but not create new ones) and MVC 5 <p>Upgrading MVC versions:</p> <ul style="list-style-type: none"> For information about how to automatically upgrade from MVC 2 to MVC 3, see ASP.NET MVC 3 Application Upgrader. For information about how to manually upgrade from MVC 2 to MVC 3, see Upgrading an ASP.NET MVC 2 Project to ASP.NET MVC 3 Tools Update. For information about how to manually upgrade from MVC3 to MVC 4, see Upgrading an ASP.NET MVC 3 Project to ASP.NET MVC 4. If your project targets .NET Framework 3.5 SP1, you must retarget it to use .NET Framework 4. For information about how to manually upgrade from MVC 4 to MVC 5, see How to Upgrade an ASP.NET MVC 4 and Web API Project to ASP.NET MVC 5 and Web API 2.
Modeling	<p>If you allow Visual Studio to update the project automatically, you can open it in Visual Studio 2015, Visual Studio 2013, or Visual Studio 2012.</p> <p>The format of the modeling project has not changed between Visual Studio 2015 and Visual Studio 2017 and the project can be opened and modified in either version. However, there are differences in behavior in Visual Studio 2017:</p> <ul style="list-style-type: none"> Modeling projects are now referred to as "Dependency Validation" projects in the menus and templates. UML diagrams are no longer supported in Visual Studio 2017. UML files are listed in the Solution Explorer as before but are opened as XML files. Use Visual Studio 2015 to view, create, or edit UML diagrams. In Visual Studio 2017, validation of architectural dependencies is no longer performed when the modeling project is built. Instead, validation is carried out as each code project is built. This change does not affect the modeling project, but it does require changes to the code projects being validated. Visual Studio 2017 can automatically make the necessary changes to the code projects (more information).
MSI Setup (vdproj)	See InstallShield Projects.
Office 2007 VSTO	Requires a one-way upgrade for Visual Studio 2017.

TYPE OF PROJECT	SUPPORT
Office 2010 VSTO	If the project targets the .NET Framework 4, you can open it in Visual Studio 2010 SP1 and later. All other projects require a one-way upgrade.
Service Fabric (sfproj)	Service Fabric Application projects can be opened in either Visual Studio 2015 or Visual Studio 2017, unless the Service Fabric Application project references an ASP.NET Core service project. Service Fabric projects from Visual Studio 2015 that are opened in Visual Studio 2017 are one-way migrated from the xproj format to csproj. See ".NET Core projects (xproj)" earlier in this table.
SharePoint 2010	<p>When a SharePoint solution project is opened with Visual Studio 2017, it's upgraded to either SharePoint 2013 or SharePoint 2016. The ".NET Desktop Development" workload must be installed in Visual Studio 2017 for the upgrade.</p> <p>For more information about how to upgrade SharePoint projects, see Upgrade to SharePoint 2013, Update Workflow in SharePoint Server 2013, and Create the SharePoint Server 2016 farm for a database attach upgrade.</p>
SharePoint 2016	SharePoint Add-In projects created in Office Developer Tools Preview 2 cannot be opened in Visual Studio 2017. To work around this limitation, update the <code>MinimumVisualStudioVersion</code> to 12.0 and <code>MinimumOfficeToolsVersion</code> to 12.2 in the csproj vbproj file.
Silverlight	Silverlight projects not supported in Visual Studio 2017. To maintain Silverlight applications, continue to use Visual Studio 2015.
SQL Server Reporting Services and SQL Server Analysis Services (SSRS, SSDT, SSAS, MSAS)	Support for these project types is provided through two extensions in the Visual Studio Gallery: Microsoft Analysis Services Modeling Projects and Microsoft Reporting Services Projects . SSDT support is also included with the Data Storage and Processing workload in Visual Studio 2017. For more information, see the Download and install SQL Server Data Tools (SSDT) for Visual Studio page.
SQL Server Integration Services (SSIS)	Support for Visual Studio 2017 is available through the SQL Server Data Tools (SSDT). For more information, see the Download and install SQL Server Data Tools (SSDT) for Visual Studio page, and the SQL Server Integration Services (SSIS) team blog.
Visual C++	You can use Visual Studio 2017 to work in projects that were created in earlier versions of Visual Studio back to Visual Studio 2010. When you first open the project, you have the option to upgrade to the latest compiler and toolset or to continue using the original ones. If you choose to keep using the original ones, Visual Studio 2017 does not modify the project file, and uses the toolset from the earlier Visual Studio installation to build your project. Keeping the original options means you can still open the project in the original version of Visual Studio if necessary. For more information, see Use native multi-targeting in Visual Studio to build old projects .

TYPE OF PROJECT	SUPPORT
Visual Studio Extensibility/VSIX	Projects with MinimumVersion 14.0 or less are updated to declare MinimumVersion 15.0, which prevents the project from being opened in earlier versions of Visual Studio. To allow a project to open in earlier versions, set MinimumVersion to <code>\$(VisualStudioVersion)</code> . See also How to: Migrate Extensibility Projects to Visual Studio 2017 .
Visual Studio Lab Management	You can use Microsoft Test Manager or Visual Studio 2010 SP1 and later to open environments created in any of these versions. However, for Visual Studio 2010 SP1 the version of Microsoft Test Manager must match the version of Team Foundation Server before you can create environments.
Visual Studio Tools for Apache Cordova	Projects can be opened in Visual Studio 2017, but are not backwards compatible. Upon opening a project from Visual Studio 2015, you're prompted to allow modifications to your project. This modification upgrades the project to use toolsets instead of a <code>taco.json</code> file to manage the versioning of the Cordova library, its platforms, its plugins, and its node/npm dependencies. See the migration guide for more information.
Web Deployment (wdproj)	Support for Web Deployment projects was removed in Visual Studio 2012 with the addition of publish profile support. Because there is no equivalent in Visual Studio 2017, there is no automatic migration path for such projects. Instead, open the wdproj file in a text editor and copy-paste any customizations into to the pubxml (publish profile) file, as described on StackOverflow .
Windows Communication Foundation, Windows Workflow Foundation	You can open this project in Visual Studio 2017, Visual Studio 2015, Visual Studio 2013, and Visual Studio 2012
Windows Presentation Foundation	You can open this project in Visual Studio 2017, Visual Studio 2013, Visual Studio 2012, and Visual Studio 2010 SP1.
Windows Store/Phone apps	Projects for Windows Store 8.1 and 8.0, and Windows Phone 8.1 and 8.0 are not supported in Visual Studio 2017. To maintain these apps, continue to use Visual Studio 2015. To maintain Windows Phone 7.x projects, use Visual Studio 2012.

How Visual Studio decides when to migrate a project

Each new version of Visual Studio generally seeks to maintain compatibility with previous versions, such that the same project can be opened, modified, and built across different versions. However, there are inevitable changes over time such that some project types may no longer be supported. (See [Platform Targeting and Compatibility](#) for which project types are supported in Visual Studio 2017.) In these cases, a newer version of Visual Studio won't load the project and doesn't offer a migration path; you need to maintain that project in a previous version of Visual Studio that does support it.

In other cases, the newer version of Visual Studio can open a project, but must update or migrate the project in such a way that might render it incompatible with previous versions. Visual Studio uses a number of criteria to determine whether such migration is necessary:

- Compatibility with the target versions of platforms, back to Visual Studio 2013 RTM.

- Compatibility of design-time assets with previous versions of Visual Studio. (Namely different channels of Visual Studio 2017; Visual Studio 2015 RTM & Update 3; Visual Studio 2013 RTM & Update 5; Visual Studio 2012 Update 4; Visual Studio 2010 SP 1.) Visual Studio 2017 aims to fail gracefully with deprecated design-time assets without corrupting them, such that previous versions can still open the project.
- Whether new design time assets would break compatibility with previous versions down to Visual Studio 2013 RTM & Update 5.

The engineering owner of the project type in question looks at these criteria and makes the call where support, compatibility, and migration are concerned. Again, Visual Studio tries to maintain transparent compatibility between Visual Studio versions if possible, meaning that one can create and modify projects in one version of Visual Studio and it just works in other versions.

If such compatibility is not possible, however, as with some of the project types described in this article, then Visual Studio opens the upgrade wizard to make the necessary one-way changes.

Such one-way changes may involve changing the `ToolsVersion` property in the project file, which denotes exactly which version of MSBuild can turn the project's source code into runnable and deployable artifacts that you ultimately want. That is, what renders a project incompatible with previous versions of Visual Studio is not the *Visual Studio* version, but the *MSBuild* version, as determined by `ToolsVersion`. So long as your version of Visual Studio contains the MSBuild toolchain that matches the `ToolsVersion` in a project, then Visual Studio can invoke that toolchain to build the project.

To maintain maximum compatibility with projects created in older versions, Visual Studio 2017 includes the necessary MSBuild toolchains to support `ToolsVersion` 15, 14, 12, and 4. Projects that use any of these `ToolsVersion` values should result in a successful build. (Subject, again, to whether Visual Studio 2017 supports the project type at all, as described on [Platform Targeting and Compatibility](#).)

In this context, the question naturally arises whether you should try to manually update or migrate a project to a newer `ToolsVersion` value. Making such a change is unnecessary, and would likely generate many errors and warnings that you'd need to fix to get the project to build again. Furthermore, if Visual Studio drops support for a specific `ToolsVersion` in the future, then opening the project will trigger the project migration process specifically because the `ToolsVersion` value must be changed. In such a case, the subsystem for that specific project type knows exactly what needs to be changed, and can make those changes automatically as described earlier in this article.

Next steps

Refer to the following articles for further discussion:

- [ToolsVersion guidance](#)
- [Framework targeting guidance](#)

See also

[Project migration and upgrade reference for Visual Studio 2019](#)

Each new version of Visual Studio generally supports most previous types of projects, files, and other assets. You can work with them [as you always have](#), and provided that you don't depend on newer features, Visual Studio tries to preserve backwards compatibility with previous versions like Visual Studio 2017, Visual Studio 2015, Visual Studio 2013, and Visual Studio 2012. (See the [Release Notes](#) for which features are specific to which versions.)

Support for some project types changes over time. A newer version of Visual Studio might no longer support certain projects at all, or it might require updating a project so that it's no longer backwards-compatible. For

current status on migration issues, refer to the [Visual Studio Developer Community](#).

This article provides details for project types that Visual Studio 2019 can migrate. It also includes information about project types that are deprecated in Visual Studio 2019, or are soon to be deprecated. The article excludes project types that are no longer supported in Visual Studio 2019 and cannot therefore be migrated. The article also excludes supported project types that have no migration issues; that list is found on [Platform Targeting and Compatibility](#).

IMPORTANT

Certain project types require that you install specific workloads through the Visual Studio Installer. If you don't have the workload installed, Visual Studio reports an unknown or incompatible project type. In that case, check your installation options and try again. See the [Platform Targeting and Compatibility](#) article for details on project support in Visual Studio 2019.

Project types

The following list describes support in Visual Studio 2019 for projects that were created in earlier versions.

If you don't see a project or file type listed here that should be, consult the [Visual Studio 2017 version of this article](#). You can also use the **Send feedback about > This page** button at the bottom of this page to provide details of your project. (If you use the anonymous "Is this page helpful?" control, we aren't able to respond to your feedback.)

TYPE OF PROJECT	SUPPORT
.NET Core projects (xproj)	<p>Projects created with Visual Studio 2015 used preview tooling that included an xproj project file.</p> <p>Visual Studio 2017: The xproj format is not supported other than for migration to csproj format. When you open an xproj file, you're prompted to migrate the file to the SDK-style csproj format. (A backup of the xproj file is made.) SDK-style csproj projects are not supported in Visual Studio 2015 and earlier.</p> <p>Visual Studio 2019: In version 16.3 and later, you cannot load or migrate xproj projects. For more information, see Migrating .NET Core projects to the csproj format.</p>
ASP.NET Web Application and ASP.NET Core Web Application with Application Insights enabled	<p>For each Visual Studio user, resource information is stored in the registry per user instance. This information is used when a user doesn't have a project opened and wants to search Azure Application Insights data. Visual Studio 2015 uses different registry location than Visual Studio 2017 and Visual Studio 2019 and does not conflict.</p> <p>Once a user creates an ASP.NET Web Application or ASP.NET Core Web Application, the resource is stored in the .suo file. The user can open the project in Visual Studio 2015, Visual Studio 2017, or Visual Studio 2019, and the resource information is used for each as long as Visual Studio supports projects and solutions being used across both versions. Users need to authenticate once on each product. For example, if a project is created with Visual Studio 2017 and opened in Visual Studio 2019, the user needs to authenticate on Visual Studio 2019.</p>

TYPE OF PROJECT	SUPPORT
C#/Visual Basic Webform or Windows Form	You can open the project in Visual Studio 2019, Visual Studio 2017, and Visual Studio 2015.
Coded UI Test	<p>Coded UI test for automated UI-driven functional testing is deprecated in Visual Studio 2019.</p> <p>Visual Studio 2019 will be the last release for Coded UI test. We recommend using Selenium for testing web apps and Appium with WinAppDriver for testing desktop and UWP apps.</p>
Database Unit Test Projects (.csproj, .vbproj)	Older Data Unit test projects are loaded in Visual Studio 2019 but use the GAC'd version of dependencies. To upgrade the unit test project to use the latest dependencies, right-click on the project in Solution Explorer and select Convert to SQL Server Unit Testing Project... .
F#	Visual Studio 2019 can open projects created in Visual Studio 2013, Visual Studio 2015, and Visual Studio 2017. A key difference from older Visual Studio templates for new projects is that the FSharp.Core version is now always a NuGet package. F# is installed by default with any .NET Workload.
InstallShield MSI setup	Installer projects created in Visual Studio 2010 can be opened in later versions with the help of the Visual Studio Installer Projects extension . Also see the WiX Toolset Visual Studio 2017 Extension . InstallShield Limited Edition is no longer included with Visual Studio. Check with Flexera Software about availability for Visual Studio 2019.
LightSwitch	LightSwitch is no longer supported in Visual Studio 2019 or in Visual Studio 2017. Projects created with Visual Studio 2012 and earlier opened in Visual Studio 2013 or Visual Studio 2015 are upgraded and can be opened only in Visual Studio 2013 or Visual Studio 2015 thereafter.
Load Test	<p>Web performance and load test capabilities are deprecated in Visual Studio 2019.</p> <p>Visual Studio 2019 will be the last release for load test. Use alternative load testing tools such as Apache JMeter, Akamai CloudTest, Blazemeter.</p>
Microsoft Azure Tools for Visual Studio	To open these types of projects, first install the Azure SDK for .NET , then open the project. If necessary, your project is updated.
Microsoft Test Manager	<p>Microsoft Test Manager and Feedback Client are no longer shipping in Visual Studio, starting with Visual Studio 2019.</p> <p>Leverage Azure Test Plans (part of Azure DevOps) for your manual and exploratory testing needs. For more information, see the Guidance on Microsoft Test Manager usage page in the Azure DevOps documentation.</p>

TYPE OF PROJECT	SUPPORT
Model-View-Controller framework (ASP.NET MVC)	<p>Support for MVC versions and Visual Studio:</p> <ul style="list-style-type: none"> Visual Studio 2010 SP1 supports MVC 2 and MVC 3; MVC 4 support is added through the ASP.NET 4 MVC 4 for Visual Studio 2010 SP1 download Visual Studio 2012 supports only MVC 3 and MVC 4 Visual Studio 2013 supports only MVC 4 and MVC 5 Visual Studio 2019, Visual Studio 2017, and Visual Studio 2015 support MVC 4 (you can open existing projects but not create new ones) and MVC 5 <p>Upgrading MVC versions:</p> <ul style="list-style-type: none"> For information about how to automatically upgrade from MVC 2 to MVC 3, see ASP.NET MVC 3 Application Upgrader. For information about how to manually upgrade from MVC 2 to MVC 3, see Upgrading an ASP.NET MVC 2 Project to ASP.NET MVC 3 Tools Update. For information about how to manually upgrade from MVC3 to MVC 4, see Upgrading an ASP.NET MVC 3 Project to ASP.NET MVC 4. If your project targets .NET Framework 3.5 SP1, you must retarget it to use .NET Framework 4. For information about how to manually upgrade from MVC 4 to MVC 5, see How to Upgrade an ASP.NET MVC 4 and Web API Project to ASP.NET MVC 5 and Web API 2.
Modeling	<p>If you allow Visual Studio to update the project automatically, you can open it in Visual Studio 2015, Visual Studio 2013, or Visual Studio 2012.</p> <p>The format of the modeling project has not changed since Visual Studio 2015 and the project can be opened and modified in these versions. However, there are differences in behavior in Visual Studio 2017 and Visual Studio 2019:</p> <ul style="list-style-type: none"> Modeling projects are now referred to as "Dependency Validation" projects in the menus and templates. UML diagrams are no longer supported in Visual Studio 2017 and Visual Studio 2019. UML files are listed in the Solution Explorer as before but are opened as XML files. Use Visual Studio 2015 to view, create, or edit UML diagrams. In Visual Studio 2019, validation of architectural dependencies is no longer performed when the modeling project is built. Instead, validation is carried out as each code project is built. This change does not affect the modeling project, but it does require changes to the code projects being validated. Visual Studio 2019 can automatically make the necessary changes to the code projects.
MSI Setup (vdproj)	See InstallShield Projects.
Office 2007 VSTO	Requires a one-way upgrade for Visual Studio 2019.

TYPE OF PROJECT	SUPPORT
Office 2010 VSTO	If the project targets the .NET Framework 4, you can open it in Visual Studio 2010 SP1 and later. All other projects require a one-way upgrade.
Portable Class Library (PCL)	<p>Portable Class Libraries (or PCLs) are now unsupported. Visual Studio 2019 will still open and build them, but it is not possible to create new PCL projects. We recommend migrating code in a PCL project to a .NET Standard project.</p> <p>PCL support will no longer be included by default, but will be available on the Visual Studio "Individual Components" tab.</p>
Python Workload	<p>Support for Python Windows IoT Core apps was removed in Visual Studio 2019. Because there is no equivalent in Visual Studio 2019 Preview, there is no automatic migration path for such projects.</p> <p>You can continue using Visual Studio 2017.</p>
R Tools for Visual Studio	<p>R Tools for Visual Studio was removed from the Data Science Workload in Visual Studio 2019.</p> <p>You can continue using Visual Studio 2017 or alternatives like RStudio.</p>
Service Fabric (sfproj)	Service Fabric Application projects can be opened in either Visual Studio 2015, Visual Studio 2017, and Visual Studio 2019 Preview, unless the Service Fabric Application project references an ASP.NET Core service project. Service Fabric projects from Visual Studio 2015 that are opened in Visual Studio 2017 or in Visual Studio 2019 Preview are one-way migrated from the xproj format to csproj. See ".NET Core projects (xproj)" earlier in this table.
SharePoint 2010	<p>When a SharePoint solution project is opened with Visual Studio 2019, it's upgraded to either SharePoint 2013 or SharePoint 2016. The ".NET Desktop Development" workload must be installed in Visual Studio 2019 for the upgrade.</p> <p>For more information about how to upgrade SharePoint projects, see Upgrade to SharePoint 2013, Update Workflow in SharePoint Server 2013, and Create the SharePoint Server 2016 farm for a database attach upgrade.</p>
SharePoint 2016	SharePoint Add-In projects created in Office Developer Tools Preview 2 cannot be opened in Visual Studio 2019. To work around this limitation, update the <code>MinimumVisualStudioVersion</code> to 12.0 and <code>MinimumOfficeToolsVersion</code> to 12.2 in the csproj vbproj file.
Silverlight	Silverlight projects not supported in Visual Studio 2019. To maintain Silverlight applications, continue to use Visual Studio 2015.

TYPE OF PROJECT	SUPPORT
SQL - Redgate	<p>Redgate's SQL Change Automation Core (previously called ReadyRoll Core), SQL Prompt Core, and SQL Search are no longer shipping in the Visual Studio installer.</p> <p>You can continue using Visual Studio 2017 for these features. In Visual Studio 2019, you can upgrade to the paid SQL Change Automation and SQL Prompt products that are available in Redgate's SQL Toolbelt.</p>
SQL Server Reporting Services and SQL Server Analysis Services (SSRS, SSDT, SSAS, MSAS)	<p>Support for these project types is provided through two extensions in the Visual Studio Gallery: Microsoft Analysis Services Modeling Projects and Microsoft Reporting Services Projects. SSDT support is also included with the Data Storage and Processing workload in Visual Studio 2019. For more information, see the Download and install SQL Server Data Tools (SSDT) for Visual Studio page.</p>
SQL Server Integration Services (SSIS)	<p>Support for Visual Studio 2019 is available. For more information, see the Download and install SQL Server Data Tools (SSDT) for Visual Studio page, the SQL Server Integration Services (SSIS) team blog, and the SQL Server Integration Services Projects page on the Marketplace.</p>
Test Window Extension	<p>In Visual Studio 2019, some test window APIs that were previously marked public but were never officially documented have been removed. Widely visible APIs were marked deprecated in Visual Studio 2017 to give extension maintainers an early warning. To our knowledge, few extensions have taken a dependency on these APIs. For more info and updates, view the complete list of deprecated test-related APIs. If this affects your scenario, please let us know on developer community.</p>
Visual C++	<p>You can use Visual Studio 2019 to work in projects that were created in earlier versions of Visual Studio back to Visual Studio 2010. When you first open the project, you have the option to upgrade to the latest compiler and toolset or to continue using the original ones. If you choose to keep using the original ones, Visual Studio 2019 does not modify the project file, and uses the toolset from the earlier Visual Studio installation to build your project. Keeping the original options means you can still open the project in the original version of Visual Studio if necessary. For more information, see Use native multi-targeting in Visual Studio to build old projects.</p>
Visual Studio Extensibility/VSIX	<p>Projects with MinimumVersion 14.0 or less are updated to declare MinimumVersion 15.0, which prevents the project from being opened in earlier versions of Visual Studio. To allow a project to open in earlier versions, set MinimumVersion to <code>\$(VisualStudioVersion)</code>. See also How to: Migrate Extensibility Projects to Visual Studio 2017.</p>
Visual Studio Lab Management	<p>You can use Microsoft Test Manager or Visual Studio 2010 SP1 and later to open environments created in any of these versions. However, for Visual Studio 2010 SP1 the version of Microsoft Test Manager must match the version of Team Foundation Server before you can create environments.</p>

TYPE OF PROJECT	SUPPORT
Visual Studio Tools for Apache Cordova	<p>Support for Apache Cordova was removed in Visual Studio 2019. Because there is no equivalent in Visual Studio 2019, there is no automatic migration path for such projects.</p> <p>You can use the Cordova Tools for Visual Studio Code extension (which provides support for the latest version of Cordova) or continue using Visual Studio 2017.</p>
Web Deployment (wdproj)	<p>Support for Web Deployment projects was removed in Visual Studio 2012 with the addition of publish profile support. Because there is no equivalent in Visual Studio 2019, there is no automatic migration path for such projects. Instead, open the wdproj file in a text editor and copy-paste any customizations into the pubxml (publish profile) file, as described on StackOverflow.</p>
Windows Communication Foundation, Windows Workflow Foundation	<p>You can open this project in Visual Studio 2019, Visual Studio 2017, Visual Studio 2015, Visual Studio 2013, and Visual Studio 2012.</p>
Windows Presentation Foundation	<p>You can open this project in Visual Studio 2019, Visual Studio 2017, Visual Studio 2013, Visual Studio 2012, and Visual Studio 2010 SP1.</p>
Windows Phone apps	<p>Projects for Windows Phone are not supported in Visual Studio 2019.</p> <p>To maintain Windows Phone 8.x apps, use Visual Studio 2015. To maintain Windows Phone 7.x projects, use Visual Studio 2012.</p>
Windows Store apps	<p>JavaScript Universal Windows Projects are not supported in Visual Studio 2019. To maintain these projects, use Visual Studio 2017.</p> <p>Windows 10 SDKs before the Windows 10 Fall Creators Update (build 16299) have been removed from the Visual Studio 2019 installer. You can download the older SDKs manually or retarget your projects to use the newer SDKs.</p> <p>Universal Windows Projects using project.json are not supported. We recommend upgrading these projects to use package references. Alternately, add a reference to Microsoft.NET.Test.Sdk version 16.0.0.0 in the project.json file.</p> <p>Projects for Windows Store 8.1 and 8.0 are not supported in Visual Studio 2019. To maintain these apps, continue to use Visual Studio 2015.</p>
Xamarin	<p>The Xamarin Live Player extension for Visual Studio and Visual Studio for Mac has been removed. This removes the pairing screen and any integration. Instead, use the built in Xamarin.Forms Previewer.</p> <p>The Visual Studio Emulator for Android has been removed from the Visual Studio Installer. Instead, use the new Hyper-V support in the Google Android emulator.</p>

How Visual Studio decides when to migrate a project

Each new version of Visual Studio generally seeks to maintain compatibility with previous versions, such that the same project can be opened, modified, and built across different versions. However, there are inevitable changes over time such that some project types may no longer be supported. (See [Platform Targeting and Compatibility](#) for which project types are supported in Visual Studio 2019.) In these cases, a newer version of Visual Studio won't load the project and doesn't offer a migration path; you need to maintain that project in a previous version of Visual Studio that does support it.

In other cases, the newer version of Visual Studio can open a project, but must update or migrate the project in such a way that might render it incompatible with previous versions. Visual Studio uses a number of criteria to determine whether such migration is necessary:

- Compatibility with the target versions of platforms, back to Visual Studio 2013 RTM.
- Compatibility of design-time assets with previous versions of Visual Studio. (Namely different channels of Visual Studio 2019, Visual Studio 2017; Visual Studio 2015 RTM & Update 3; Visual Studio 2013 RTM & Update 5; Visual Studio 2012 Update 4; Visual Studio 2010 SP 1.) Visual Studio 2019 aims to fail gracefully with deprecated design-time assets without corrupting them, such that previous versions can still open the project.
- Whether new design time assets would break compatibility with previous versions down to Visual Studio 2013 RTM & Update 5.

The engineering owner of the project type in question looks at these criteria and makes the call where support, compatibility, and migration are concerned. Again, Visual Studio tries to maintain transparent compatibility between Visual Studio versions if possible, meaning that one can create and modify projects in one version of Visual Studio and it just works in other versions.

If such compatibility is not possible, however, as with some of the project types described in this article, then Visual Studio opens the upgrade wizard to make the necessary one-way changes.

Such one-way changes may involve changing the `ToolsVersion` property in the project file, which denotes exactly which version of MSBuild can turn the project's source code into runnable and deployable artifacts that you ultimately want. That is, what renders a project incompatible with previous versions of Visual Studio is not the *Visual Studio* version, but the *MSBuild* version, as determined by `ToolsVersion`. So long as your version of Visual Studio contains the MSBuild toolchain that matches the `ToolsVersion` in a project, then Visual Studio can invoke that toolchain to build the project.

To maintain maximum compatibility with projects created in older versions, Visual Studio 2019 includes the necessary MSBuild toolchains to support `ToolsVersion` 15, 14, 12, and 4. Projects that use any of these `ToolsVersion` values should result in a successful build. (Subject, again, to whether Visual Studio 2019 supports the project type at all, as described on [Platform Targeting and Compatibility](#).)

In this context, the question naturally arises whether you should try to manually update or migrate a project to a newer `ToolsVersion` value. Making such a change is unnecessary, and would likely generate many errors and warnings that you'd need to fix to get the project to build again. Furthermore, if Visual Studio drops support for a specific `ToolsVersion` in the future, then opening the project will trigger the project migration process specifically because the `ToolsVersion` value must be changed. In such a case, the subsystem for that specific project type knows exactly what needs to be changed, and can make those changes automatically as described earlier in this article.

Next steps

Refer to the following articles for further discussion:

- [ToolsVersion guidance](#)
- [Framework targeting guidance](#)

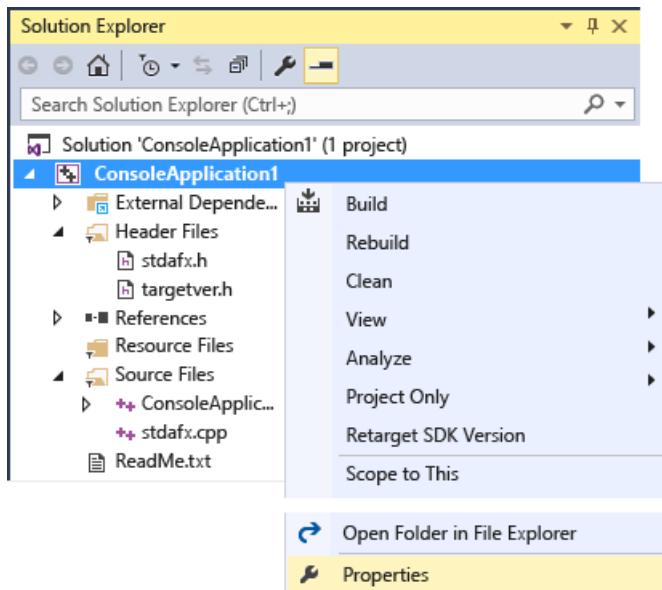
See also

[Project migration and upgrade reference for Visual Studio 2017](#)

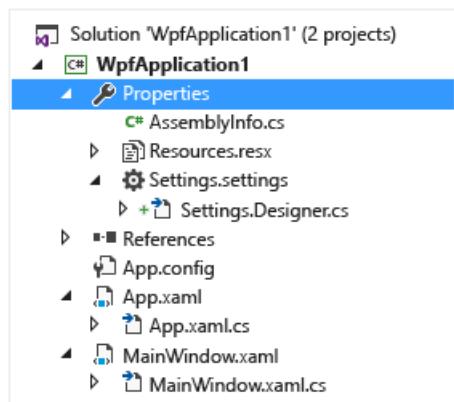
Manage project and solution properties

10/18/2019 • 2 minutes to read • [Edit Online](#)

Projects have properties that govern many aspects of compilation, debugging, testing and deploying. Some properties are common among all project types, and some are unique to specific languages or platforms. You access project properties by right-clicking the project node in **Solution Explorer** and choosing **Properties** or by typing **properties** into the search box on the menu bar and choosing **Properties Window** from the results.



.NET projects might also have a properties node in the project tree itself.



NOTE

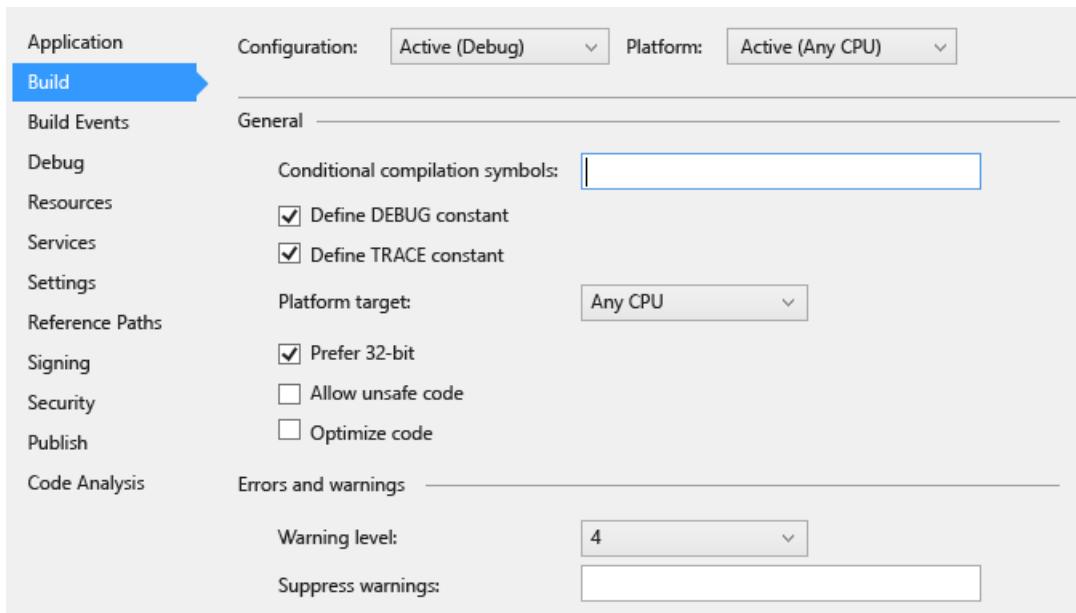
This topic applies to Visual Studio on Windows. For Visual Studio for Mac, see [Managing solution and project properties \(Visual Studio for Mac\)](#).

Project properties

Project properties are organized into groups, and each group has its own property page. The pages might be different for different languages and project types.

C#, Visual Basic, and F# projects

In C#, Visual Basic, and F# projects, properties are exposed in the **Project Designer**. The following illustration shows the **Build** property page for a WPF project in C#:



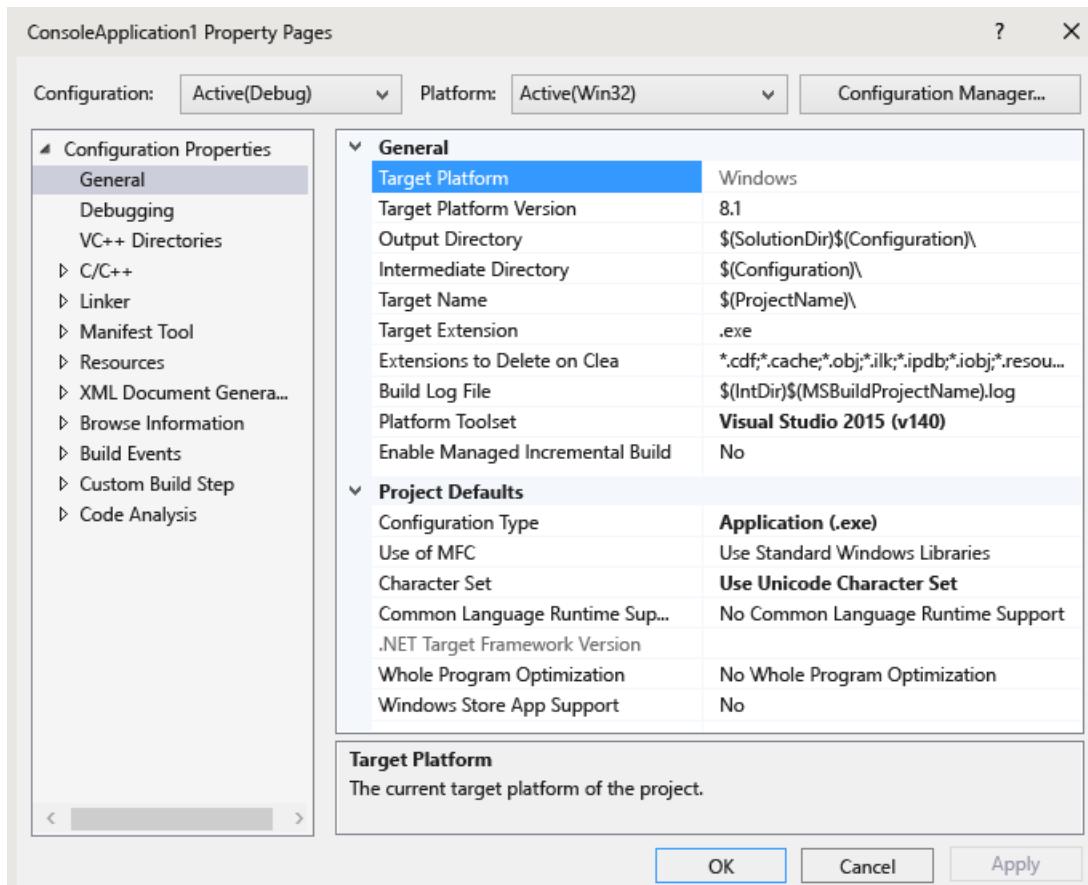
For information about each of the property pages in **Project Designer**, see [Project properties reference](#).

TIP

Solutions have a few properties, and so do project items; these properties are accessed in the [Properties window](#), not **Project Designer**.

C++ and JavaScript projects

C++ and JavaScript projects have a different user interface for managing project properties. This illustration shows a C++ project property page (JavaScript pages are similar):



For information about C++ project properties, see [Work with project properties \(C++\)](#). For more information

about JavaScript properties, see [Property pages, JavaScript](#).

Solution properties

To access properties on the solution, right click the solution node in **Solution Explorer** and choose **Properties**. In the dialog, you can set project configurations for **Debug** or **Release** builds, choose which projects should be the startup project when **F5** is pressed, and set code analysis options.

See also

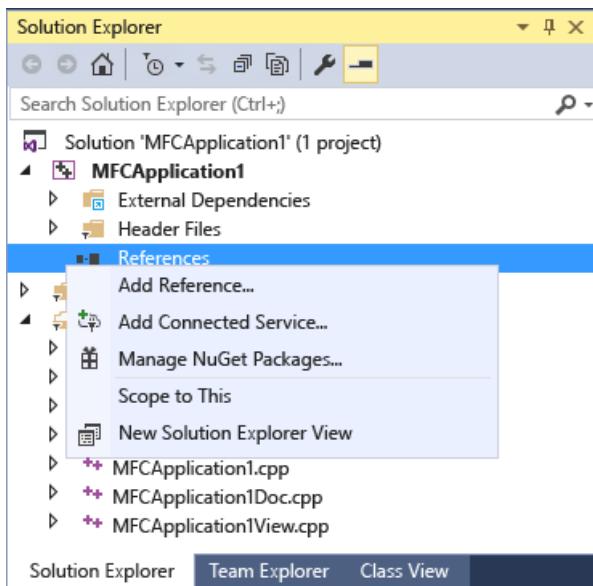
- [Solutions and projects in Visual Studio](#)
- [Managing solution and project properties \(Visual Studio for Mac\)](#)

Manage references in a project

10/18/2019 • 6 minutes to read • [Edit Online](#)

Before you write code against an external component or connected service, your project must first contain a reference to it. A reference is essentially an entry in a project file that contains the information that Visual Studio needs to locate the component or the service.

To add a reference, right click on the **References** or **Dependencies** node in **Solution Explorer** and choose **Add Reference**. You can also right-click on the project node and select **Add > Reference**. For more information, see [How to: Add or remove references](#).



You can add a reference to the following types of components and services:

- .NET class libraries or assemblies
- UWP apps
- COM components
- Other assemblies or class libraries of projects in the same solution
- Shared projects
- XML Web services

UWP app references

Project references

Universal Windows Platform (UWP) projects can create references to other UWP projects in the solution, or to Windows 8.1 projects or binaries, provided that these projects do not use APIs that have been deprecated in Windows 10. For more information, see [Move from Windows Runtime 8 to UWP](#).

If you choose to retarget Windows 8.1 projects to Windows 10, see [Port, migrate, and upgrade Visual Studio projects](#).

Extension SDK references

Visual Basic, C#, C++ and JavaScript Universal Windows Platform (UWP) apps can reference Extension SDKs

that target Windows 8.1, as long as these Extension SDKs do not use APIs that have been deprecated in Windows 10. Please check the Extension SDK vendor site to find out whether it can be referenced by UWP apps.

If you determine that the Extension SDK being referenced by your app is not supported, then you need to perform the following steps:

1. Look at the name of the project that is causing the error. The platform your project is targeting is noted in parentheses next to the project name. For example, **MyProjectName (Windows 8.1)** means that your project **MyProjectName** is targeting platform version Windows 8.1.
2. Go to the site of the vendor who owns the unsupported Extension SDK, and install the version of the Extension SDK with dependencies that are compatible with the version of the platform your project is targeting.

NOTE

One way to find out whether an Extension SDK has dependencies on other Extension SDKs is by looking in **Reference Manager**. Restart Visual Studio, create a new C# UWP app project, and then right-click on the project and choose **Add Reference**. Go to the **Windows** tab, then the **Extensions** sub-tab, and select the Extension SDK. Look at the right pane in the **Reference Manager**. If it has dependencies, they will be listed there.

IMPORTANT

If your project is targeting Windows 10, and the Extension SDK installed in the previous step has a dependency on the Microsoft Visual C++ Runtime Package, the version of Microsoft Visual C++ Runtime Package that is compatible with Windows 10 is v14.0, and is installed with Visual Studio.

3. If the Extension SDK you installed in the previous step has dependencies on other Extension SDKs, go to the sites of the vendors who own the dependencies, and install the versions of these dependencies that are compatible with the version of the platform your project is targeting.
4. Restart Visual Studio and open your app.
5. Right-click on the **References** or **Dependencies** node in the project that caused the error and choose **Add Reference**.
6. Click the **Windows** tab and then the **Extensions** sub-tab, then uncheck the checkboxes for the old Extension SDKs, and check the checkboxes for the new Extension SDKs. Click **OK**.

Add a reference at design time

When you make a reference to an assembly in your project, Visual Studio searches for the assembly in the following locations:

- The current project directory. (You can find these assemblies by using the **Browse** tab.)
- Other project directories in the same solution. (You can find these assemblies on the **Projects** tab.)

NOTE

- All projects contain an implied reference to **mscorlib**.
- All projects contain an implied reference to **System.Core**, even if **System.Core** is removed from the list of references.
- Visual Basic projects contain an implied reference to **Microsoft.VisualBasic**.

References to shared components at run time

At run time, components must be either in the output path of the project or in the Global Assembly Cache (GAC). If the project contains a reference to an object that is not in one of these locations, you must copy the reference to the output path of the project when you build the project. The [CopyLocal](#) property indicates whether this copy has to be made. If the value is **True**, the reference is copied to the project directory when you build the project. If the value is **False**, the reference is not copied.

If you deploy an application that contains a reference to a custom component that is registered in the GAC, the component will not be deployed with the application, regardless of the [CopyLocal](#) setting. In earlier versions of Visual Studio, you could set the [CopyLocal](#) property on a reference to ensure that the assembly was deployed. Now, you must manually add the assembly to the \Bin folder. This puts all custom code under scrutiny, reducing the risk of publishing custom code with which you are not familiar.

By default, the [CopyLocal](#) property is set to **False** if the assembly or component is in the global assembly cache or is a framework component. Otherwise, the value is set to **True**. Project-to-project references are always set to **True**.

Reference a project or assembly that targets a different version of .NET

You can create applications that reference projects or assemblies that target a different version of the .NET. For example, you could create an application that targets .NET Framework 4.6, that references an assembly that targets .NET Framework 4.5. If you create a project that targets an earlier version of .NET, you cannot set a reference in that project to a project or assembly that targets a newer version.

For more information, see [Framework targeting overview](#).

Project-to project references

Project-to-project references are references to projects that contain assemblies; you add project references by using the **Projects** tab of the Reference Manager dialog box. Visual Studio can find an assembly when given a path to the project.

When you have a project that produces an assembly, you should reference the project and not use a file reference (see below). The advantage of a project-to-project reference is that it creates a dependency between the projects in the build system. The dependent project will be built if it has changed since the last time the referencing project was built. A file reference does not create a build dependency, so it is possible to build the referencing project without building the dependent project, and the reference can become obsolete. (That is, the project can reference a previously built version of the project.) This can result in several versions of a single DLL being required in the *bin* directory, which is not possible. When this conflict occurs, you will see a message such as "Warning: the dependency 'file' in project 'project' cannot be copied to the run directory because it would overwrite the reference 'file.'". For more information, see [Troubleshoot broken references](#) and [How to: Create and remove project dependencies](#).

NOTE

A file reference instead of a project-to-project reference is created if the target version of the .NET Framework of one project is version 4.5, and the target version of the other project is version 2, 3, 3.5, or 4.0.

Shared project references

Unlike most other project types, a *shared project* does not have any binary output. Instead, the code is compiled into each project that references it. [Shared Projects](#) let you write common code that's referenced by a number of different application projects. The code is compiled as part of each referencing project and can include compiler

directives to help incorporate platform-specific functionality into the shared code base. Add a reference to a shared project on the **Shared Projects** tab of the Reference Manager dialog box.

File references

File references are direct references to assemblies outside the context of a Visual Studio project. You create them by using the **Browse** tab of the Reference Manager dialog box. Use a file reference when you just have an assembly or component, and not the project that creates it as output.

See also

- [Troubleshoot broken references](#)
- [How to: Add or remove references](#)

How to: Add or remove references by using the Reference Manager

10/18/2019 • 9 minutes to read • [Edit Online](#)

You can use the Reference Manager dialog box to add and manage references to components that you, Microsoft, or another company developed. If you're developing a Universal Windows app, your project automatically references all of the correct Windows SDK DLLs. If you are developing a .NET application, your project automatically references *mscorlib.dll*. Some .NET APIs are exposed in components that you have to add manually. References to COM components or custom components have to be added manually.

Reference Manager dialog box

The Reference Manager dialog box shows different categories on the left side, depending on the project type:

- **Assemblies**, with **Framework** and **Extensions** subgroups
- **COM** lists all COM components that are available for referencing
- **Projects**
- **Shared Projects**
- **Windows**, with **Core** and **Extensions** subgroups. You can explore the references in the Windows SDK or extension SDKs by using the **Object Browser**.
- **Browse**, with **Recent** subgroup

Add a reference

1. In **Solution Explorer**, right-click on the **References** or **Dependencies** node and choose **Add Reference**.

You can also right-click on the project node and select **Add > Reference**.

Reference Manager opens and lists the available references by group.

2. Specify the references to add, and then select **OK**.

Assemblies tab

The **Assemblies** tab lists all .NET assemblies that are available for referencing. The **Assemblies** tab doesn't list any assemblies from the global assembly cache (GAC) because assemblies in the GAC are part of the run-time environment. If you deploy or copy an application that contains a reference to an assembly that's registered in the GAC, the assembly won't be deployed or copied with the application, regardless of the **Copy Local** setting. For more information, see [Manage references in a project](#).

When you manually add a reference to any of the EnvDTE namespaces ([EnvDTE](#), [EnvDTE80](#), [EnvDTE90](#), [EnvDTE90a](#), or [EnvDTE100](#)), set the **Embed Interop Types** property of the reference to **False** in the **Properties** window. Setting this property to **True** can cause build issues because of certain EnvDTE properties that can't be embedded.

All desktop projects contain an implicit reference to **mscorlib**. Visual Basic projects contain an implicit reference to **Microsoft.VisualBasic**. All projects contain an implicit reference to **System.Core**, even if it's removed from the list of references.

If a project type doesn't support assemblies, the tab won't appear in the Reference Manager dialog box.

The **Assemblies** tab consists of two sub-tabs:

1. **Framework** lists all assemblies that constitute the targeted framework.

For projects that don't target .NET Core or the Universal Windows Platform, the **Framework** tab enumerates assemblies from the targeted framework. The user must add any references that the application requires.

Universal Windows projects contain references to all of the assemblies in the targeted framework by default. In managed projects, a read-only node under the **References** folder in **Solution Explorer** indicates the reference to the entire framework. Accordingly, the **Framework** tab doesn't enumerate any of the assemblies from the framework and instead displays the following message: "All of the Framework assemblies are already referenced. Please use the Object Browser to explore the references in the Framework".

2. **Extensions** lists all assemblies that external vendors of components and controls have developed to extend the targeted framework. Depending on the purpose of the user application, it might need these assemblies.

Extensions is populated by enumerating the assemblies that are registered in the following locations:

32-bit machine:

- HKEY_CURRENT_USER\SOFTWARE\Microsoft\[Target Framework Identifier]\v[Target Framework Version]\AssemblyFoldersEx\[UserComponentName]\@default=[Disk location of assemblies]
- HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\[Target Framework Identifier]\v[Target Framework Version]\AssemblyFoldersEx\[UserComponentName]\@default=[Disk location of assemblies]

64-bit machine:

- HKEY_CURRENT_USER\SOFTWARE\Wow6432Node\Microsoft\[Target Framework Identifier]\v[Target Framework Version]\AssemblyFoldersEx\[UserComponentName]\@default=[Disk location of assemblies]
- HKEY_LOCAL_MACHINE\SOFTWARE\Wow6432Node\Microsoft\[Target Framework Identifier]\v[Target Framework Version]\AssemblyFoldersEx\[UserComponentName]\@default=[Disk location of assemblies]

And older versions of the [Target Framework Identifier]

For example, if a project targets .NET Framework 4 on a 32-bit machine, **Extensions** enumerates assemblies that are registered under `\Microsoft.NETFramework\v4.0\AssemblyFoldersEx`, `\Microsoft.NETFramework\v3.5\AssemblyFoldersEx`, `\Microsoft.NETFramework\v3.0\AssemblyFoldersEx`, and `\Microsoft.NETFramework\v2.0\AssemblyFoldersEx`.

Some components in the list may not be shown, depending on the framework version of your project. This can occur under the following conditions:

- A component that uses a recent framework version is incompatible with a project that targets an earlier version.

For information about how to change the target framework version for a project, see [Framework targeting overview](#).

- A component that uses .NET Framework 4 is incompatible with a project that targets the .NET Framework 4.5.

You should avoid adding file references to outputs of another project in the same solution, because doing this may cause compilation errors. Instead, use the **Projects** tab of the **Add Reference** dialog box to create project-to-project references. This makes team development easier by enabling better management of the class libraries you create in your projects. For more information, see [Troubleshoot broken references](#).

NOTE

In Visual Studio 2015 or later, a file reference instead of a project reference is created if the target framework version of one project is .NET Framework 4.5 or later, and the target version of the other project is .NET Framework 2, 3, 3.5, or 4.0.

To display an assembly in the Add Reference dialog box

- Move or copy the assembly to one of the following locations:
 - The current project directory. (You can find these assemblies by using the **Browse** tab.)
 - Other project directories in the same solution. (You can find these assemblies by using the **Projects** tab.)
- or -
- Set a registry key that specifies the location of assemblies to display:

For a 32-bit operating system, add one of the following registry keys.

- [HKEY_CURRENT_USER\SOFTWARE\Microsoft\.NETFramework\
 <VersionMinimum>\AssemblyFoldersEx\MyAssemblies]@=<AssemblyLocation>"
- [HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\.NETFramework\
 <VersionMinimum>\AssemblyFoldersEx\MyAssemblies]@=<AssemblyLocation>"

For a 64-bit operating system, add one of the following registry keys in a 32-bit registry hive.

- [HKEY_CURRENT_USER\SOFTWARE\Wow6432Node\Microsoft\.NETFramework\
 <VersionMinimum>\AssemblyFoldersEx\MyAssemblies]@=<AssemblyLocation>"
- [HKEY_LOCAL_MACHINE\SOFTWARE\Wow6432Node\Microsoft\.NETFramework\
 <VersionMinimum>\AssemblyFoldersEx\MyAssemblies]@=<AssemblyLocation>"

<VersionMinimum> is the lowest framework version that applies. If <VersionMinimum> is v3.0, folders specified in *AssemblyFoldersEx* apply to projects that target .NET Framework 3.0 and later.

<AssemblyLocation> is the directory of the assemblies that you want to appear in the **Add Reference** dialog box, for example, C:\MyAssemblies.

Creating the registry key under the `HKEY_LOCAL_MACHINE` node allows all users to see the assemblies in the specified location in the **Add Reference** dialog box. Creating the registry key under the `HKEY_CURRENT_USER` node affects only the setting for the current user.

Open the **Add Reference** dialog box again. The assemblies should appear on the **.NET** tab. If they do not, make sure that the assemblies are located in the specified *AssemblyLocation* directory, restart Visual Studio, and try again.

Projects tab

The **Projects** tab lists all compatible projects within the current solution, in the **Solution** sub-tab.

A project can reference another project that targets a different framework version. For example, you could create a project that targets the .NET Framework 4 but that references an assembly that's been built for the .NET Framework 2. However, the .NET Framework 2 project can't reference a .NET Framework 4 project. For more information, see [Framework targeting overview](#).

NOTE

A project that targets the .NET Framework 4 is incompatible with a project that targets the .NET Framework 4 Client Profile.

Shared Projects tab

Add a reference to a shared project on the **Shared Projects** tab of the Reference Manager dialog box. [Shared Projects](#) let you write common code that's referenced by a number of different application projects.

Universal Windows tab

The **Universal Windows** tab lists all SDKs that are specific to platforms on which Windows operating systems run. This tab has two subgroups: **Core** and **Extensions**.

Core subgroup

Universal Windows app projects have a reference to the Universal Windows SDK by default. Accordingly, the **Core** subgroup in the **Reference Manager** doesn't enumerate any of the assemblies from the Universal Windows SDK.

Extensions subgroup

Extensions lists the user SDKs that extend the targeted Windows platform.

An SDK is a collection of files that Visual Studio treats as a single component. In the **Extensions** tab, SDKs that apply to the project from which the Reference Manager dialog box was invoked are listed as single entries. When added to a project, all of the SDK content is consumed by Visual Studio such that the user doesn't need to take any further actions to leverage the SDK contents in IntelliSense, toolbox, designers, Object Browser, build, deployment, debugging, and packaging.

For information about how to display your SDK in the **Extensions** tab, see [Creating a Software Development Kit](#).

NOTE

If a project references an SDK that depends on another SDK, Visual Studio won't consume the second SDK unless you manually add a reference to the second SDK. When a user chooses an SDK on the **Extensions** tab, the Reference Manager dialog box helps you identify SDK dependencies by listing any dependencies in the details pane.

If a project type doesn't support extensions, this tab doesn't appear in the Reference Manager dialog box.

COM tab

The **COM** tab lists all COM components that are available for referencing. If you want to add a reference to a registered COM DLL that contains an internal manifest, unregister the DLL first. Otherwise, Visual Studio adds the assembly reference as an ActiveX control instead of as a native DLL.

If a project type doesn't support COM, the tab doesn't appear in the Reference Manager dialog box.

Browse

You can use the **Browse** button to browse for a component in the file system.

A project can reference a component that targets a different framework version. For example, you could create an application that targets .NET Framework 4.7 but references a component that targets .NET Framework 4. For more information, see [Framework targeting overview](#).

Avoid adding file references to outputs of another project in the same solution, because this tactic may cause compilation errors. Instead, use the **Solution** tab of the Reference Manager dialog box to create project-to-project references. This makes team development easier by enabling better management of the class libraries that you create in your projects. For more information, see [Troubleshoot broken references](#).

You can't browse to an SDK and add it to your project. You can only browse to a file (for example, an assembly or

.winmd) and add it to your project.

When doing a file reference to a WinMD, the expected layout is that the <FileName>.winmd, <FileName>.dll, and <FileName>.pri files are all placed alongside each other. If you reference a WinMD in the following scenarios, an incomplete set of files will be copied into the project output directory and, consequently, build and runtime failures will occur.

- **Native component:** a native project will create one WinMD for each disjoint set of namespaces and one DLL that consists of the implementation. The WinMDs will have disparate names. When referencing this native component file, MSBuild won't recognize that the dissimilarly named WinMDs make one component. Consequently, only the identically named <FileName>.dll and <FileName>.winmd will be copied, and runtime errors will occur. To work around this issue, create an extension SDK. For more information, see [Create a Software Development Kit](#).
- **Consuming controls:** at a minimum, a XAML control consists of a <FileName>.winmd, <FileName>.dll, <FileName>.pri, <XamlName>.xaml, and an <ImageName>.jpg. When the project is built, the resource files that are associated with the file reference won't get copied into the project's output directory, and only <FileName>.winmd, <FileName>.dll and <FileName>.pri will be copied. A build error is logged to inform the user that the resources <XamlName>.xaml and <ImageName>.jpg are missing. To succeed, the user will have to manually copy these resource files into the project output directory for build and debugging/runtime. To work around this issue, either create an extension SDK by following the steps in [Create a Software Development Kit](#) or edit the project file to add the following property:

```
<PropertyGroup>
    <GenerateLibraryOutput>True</GenerateLibraryOutput>
</PropertyGroup>
```

NOTE

If you add the property, the build might run slower.

Recent

Assemblies, COM, Windows, and Browse each support a **Recent** tab, which enumerates the list of components that were recently added to projects.

Search

The search bar in the Reference Manager dialog box operates over the tab that's in focus. For example, if a user types "System" in the search bar while the **Solution** tab is in focus, the search won't return any results unless the solution consists of a project name that contains "System".

See also

- [Manage references in a project](#)

How to: Add or remove imported namespaces (Visual Basic)

10/18/2019 • 2 minutes to read • [Edit Online](#)

Importing a namespace allows you to use elements from that namespace in your code without fully qualifying the element. For example, if you want to access the `Create` method in the `System.Messaging.MessageQueue` class, you can import the `System.Messaging` namespace and just refer to the element you need in code as `MessageQueue.Create`.

Imported namespaces are managed on the **References** page of the **Project Designer**. The imports you specify in this dialog box are passed directly to the compiler (*/imports*) and apply to all files in your project. Use the `Imports` statement to use a namespace in a single source code file.

To add an imported namespace

1. In **Solution Explorer**, double-click the **My Project** node for the project.
2. In the **Project Designer**, click the **References** tab.
3. In the **Imported Namespaces** list, select the check box for the namespace that you wish to add.

NOTE

In order to be imported, the namespace must be in a referenced component. If the namespace does not appear in the list, you will need to add a reference to the component that contains it. For more information, see [Managing references in a project](#).

To remove an imported namespace

1. In **Solution Explorer**, double-click the **My Project** node for the project.
2. In the **Project Designer**, click the **References** tab.
3. In the **Imported Namespaces** list, clear the check box for the namespace that you wish to remove.

User imports

User imports allow you to import a specific class within a namespace rather than the entire namespace. For example, your application might have an import for the `System.Diagnostics` namespace, but the only class within that namespace that you are interested in is the `Debug` class. You can define `Debug` as a user import, and then remove the import for `System.Diagnostics`.

If you later change your mind and decide that was really the `EventLog` class that you needed, you could enter `EventLog` as a user import and overwrite `Debug` using the update functionality.

To add a user import

1. In **Solution Explorer**, double-click the **My Project** node for the project.
2. In the **Project Designer**, click the **References** tab.
3. In the text box below the **Imported Namespaces** list, enter the full name for the namespace you wish to import, including the root namespace.
4. Click the **Add user import** button to add the namespace to the **Imported Namespaces** list.

NOTE

The **Add user import** button will be disabled if the namespace matches one already in the list; you cannot add an import twice.

To update a user import

1. In **Solution Explorer**, double-click the **My Project** node for the project.
2. In the **Project Designer**, click the **References** tab.
3. In the **Imported Namespaces** list, select the namespace you wish to change.
4. In the text box below the **Imported Namespaces** list, enter the name for the new namespace.
5. Click the **Update user import** button to update the namespace in the **Imported Namespaces** list.

See also

- [Manage references in a project](#)

Troubleshoot broken references

10/18/2019 • 3 minutes to read • [Edit Online](#)

If your application attempts to use a broken reference, an exception error is generated. The inability to find the referenced component is the primary trigger for the error, but there are several situations in which a reference can be considered broken. These instances are shown in the following list:

- The project's reference path is incorrect or incomplete.
- The file being referenced has been deleted.
- The file being referenced has been renamed.
- The network connection or authentication has failed.
- The reference is to a COM component that is not installed on the computer.

The following are remedies to these problems.

NOTE

Files in assemblies are referenced with absolute paths in the project file. Therefore, it is possible for users who work in a multideveloper environment to be missing a referenced assembly in their local environment. To avoid these errors, it is better in these cases to add project-to-project references. For more information, see [Programming with assemblies](#).

Reference path is incorrect

If projects are shared on different computers, some references might not be found when a component is located in a different directory on each computer. References are stored under the name of the component file (for example, *MyComponent*). When a reference is added to a project, the folder location of the component file (for example, *C:\MyComponents*) is appended to the **ReferencePath** project property.

When the project is opened, it attempts to locate these referenced component files by looking in the directories on the reference path. If the project is opened on a computer that stores the component in a different directory, such as *D:\MyComponents*, the reference cannot be found and an error appears in the **Task List**.

To fix this problem, you can delete the broken reference and then replace it using the **Add Reference** dialog box. Another solution is to use the **Reference Path** item in the project's property pages and modify the folders in the list to point to the correct locations. The **Reference Path** property is persisted for each user on each computer. Therefore, modifying your reference path does not affect other users of the project.

TIP

Project-to-project references do not have these problems. For this reason, use them instead of file references, if you can.

To fix a broken project reference by correcting the reference path

1. In **Solution Explorer**, right-click your project node and click **Properties**.

The **Project Designer** appears.

2. If you are using Visual Basic, select the **References** page and click the **Reference Paths** button. In the **Reference Paths** dialog box, type the path of the folder that contains the item you want to reference in the

Folder field, and then click the **Add Folder** button.

If you are using C#, select the **Reference Paths** page. In the **Folder** field, type the path of the folder that contains the item you want to reference, and then click the **Add Folder** button.

Referenced file has been deleted

It is possible that the file being referenced has been deleted and no longer exists on the drive.

To fix a broken project reference for a file that no longer exists on your drive

- Delete the reference.
- If the reference exists in another location on your computer, read it from that location.

Referenced file has been renamed

It is possible that the file being referenced has been renamed.

To fix a broken reference for a file that has been renamed

- Delete the reference, and then add a reference to the renamed file.
- If the reference exists in another location on your computer, you have to read it in from that location.

Network connection or authentication has failed

There can be many possible causes for inaccessible files: a failed network connection or a failed authentication, for example. Each cause might have a unique means of recovery; for example, you might have to contact the local administrator for access to the required resources. However, deleting the reference and fixing the code which used it is always an option.

COM component is not installed on computer

If a user has added a reference to a COM component and a second user tries to run the code on a computer that does not have this component installed, the second user will receive an error that the reference is broken.

Installing the component on the second computer will correct the error. For more information about how to use references to COM components in your projects, see [COM interoperability in .NET Framework applications](#).

See also

- [References Page, Project Designer \(Visual Basic\)](#)

Manage application resources (.NET)

10/18/2019 • 2 minutes to read • [Edit Online](#)

Resource files are files that are part of an application but are not compiled, for example icon files or audio files. Since these files are not part of the compilation process, you can change them without having to recompile your binaries. If you are planning to localize your application, you should use resource files for all the strings and other resources that need to be changed when you localize your application.

NOTE

This topic applies to Visual Studio on Windows. For Visual Studio for Mac, see [Managing app resources \(Visual Studio for Mac\)](#).

For more information about resources in .NET desktop apps, see [Resources in desktop apps](#).

Work with resources

In a managed code project, open the project properties window. You can open the properties window by either:

- Right-clicking the project node in **Solution Explorer** and selecting **Properties**
- Typing **project properties** in the **Ctrl+Q** search box
- Choosing **Alt+Enter** in **Solution Explorer**

Select the **Resources** tab. You can add a **.resx** file if your project does not contain one already, add and delete different kinds of resources, and modify existing resources.

Resources in other project types

Resources are managed differently in .NET projects than in other project types. For more information about resources in:

- Universal Windows Platform (UWP) apps, see [App resources and the Resource Management System](#)
- C++ projects, see [Work with resource files](#) and [How to: Create a resource](#)

See also

- [Resources in desktop apps \(.NET Framework\)](#)
- [Managing app resources \(Visual Studio for Mac\)](#)

Manage application settings (.NET)

10/18/2019 • 5 minutes to read • [Edit Online](#)

Application settings enable you to store application information dynamically. Settings allow you to store information on the client computer that should not be included in the application code (for example a connection string), user preferences, and other information you need at run time.

Application settings replace the dynamic properties used in earlier versions of Visual Studio.

Each application setting must have a unique name. The name can be any combination of letters, numbers, or an underscore that does not start with a number, and it cannot have spaces. The name is changed through the `Name` property.

Application settings can be stored as any data type that is serialized to XML or has a `TypeConverter` that implements `Tostring` / `FromString`. The most common types are `String`, `Integer`, and `Boolean`, but you can also store values as `Color`, `Object`, or as a connection string.

Application settings also hold a value. The value is set with the `Value` property and must match the data type of the setting.

In addition, application settings can be bound to a property of a form or control at design time.

There are two types of application settings, based on scope:

- Application-scoped settings can be used for information such as a URL for a web service or a database connection string. These values are associated with the application. Therefore, users cannot change them at run time.
- User-scoped settings can be used for information such as persisting the last position of a form or a font preference. Users can change these values at run time.

You can change the type of a setting by using the `Scope` property.

The project system stores application settings in two XML files:

- an `app.config` file, which is created at design time when you create the first application setting
- a `user.config` file, which is created at run time when the user who runs the application changes the value of any user setting.

Notice that changes in user settings are not written to disk unless the application specifically calls a method to do this.

Create application settings at design time

At design time, you can create application settings in two ways: by using the **Settings** page of the **Project Designer**, or by using the **Properties** window for a form or control, which allows you to bind a setting to a property.

When you create an application-scoped setting (for example, a database connection string, or a reference to server resources), Visual Studio saves it in `app.config` with the `<applicationSettings>` tag. (Connection strings are saved under the `<connectionStrings>` tag.)

When you create a user-scoped setting (for example, default font, home page, or window size), Visual Studio saves it in `app.config` with the `<userSettings>` tag.

IMPORTANT

When you store connection strings in *app.config*, you should take precautions to avoid revealing sensitive information, such as passwords or server paths, in the connection string.

If you take connection string information from an external source, such as a user supplying a user ID and password, you must be careful to ensure that the values that you use to construct your connection string do not contain additional connection string parameters that change the behavior of your connection.

Consider using the Protected Configuration feature to encrypt sensitive information in the configuration file. For more information, see [Protect connection information](#).

NOTE

Because there is no configuration file model for class libraries, application settings do not apply for Class Library projects. The exception is a Visual Studio Tools for Office DLL project, which can have a configuration file.

Use customized settings files

You can add customized settings files to your project for convenient management of groups of settings. Settings that are contained in a single file are loaded and saved as a unit. Storing settings in separate files for frequently used and infrequently used groups can save time in loading and saving settings.

For example, you can add a file such as *SpecialSettings.settings* to your project. While your `SpecialSettings` class is not exposed in the `My` namespace, **View Code** can read the custom settings file that contains `Partial Class SpecialSettings`.

The **Settings Designer** first searches for the *Settings.settings* file that the project system creates; this file is the default file that the **Project Designer** displays in the **Settings** tab. *Settings.settings* is located in the *My Project* folder for Visual Basic projects and in the *Properties* folder for Visual C# projects. The **Project Designer** then searches for other settings files in the project's root folder. Therefore, you should put your custom settings file there. If you add a *.settings* file elsewhere in your project, the **Project Designer** will not be able to locate it.

Access or change application settings at run time in Visual Basic

In Visual Basic projects, you can access application settings at run time by using the `My.Settings` object. On the **Settings** page, click the **View code** button to view the *Settings.vb* file. *Settings.vb* defines the `Settings` class, which enables you to handle these events on the settings class: `SettingChanging`, `PropertyChanged`, `SettingsLoaded`, and `SettingsSaving`. Notice that the `Settings` class in *Settings.vb* is a partial class that displays only the user-owned code, not the whole generated class. For more information about accessing application settings by using the `My.Settings` object, see [Access application settings \(.NET Framework\)](#).

The values of any user-scoped settings that the user changes at run time (for example, the position of a form) are stored in a *user.config* file. Notice that the default values are still saved in *app.config*.

If any user-scoped settings are changed during run time, for example in testing the application, and want to reset these settings to their default values, click the **Synchronize** button.

We strongly recommend that you use the `My.Settings` object and the default *.settings* file to access settings. This is because you can use the **Settings Designer** to assign properties to settings, and, additionally, user settings are automatically saved before application shutdown. However, your Visual Basic application can access settings directly. In that case you have to access the `MySettings` class and use a custom *.settings* file in the root of the project. You must save the user settings before ending the application, as you would do for a C# application; this is described in the following section.

Access or change application settings at run time in C#

In languages other than Visual Basic, such as C#, you must access the `Settings` class directly, as shown in the following Visual C# example.

```
Properties.Settings.Default.FirstUserSetting = "abc";
```

You must explicitly call the `Save` method of this wrapper class in order to persist the user settings. You usually do this in the `Closing` event handler of the main form. The following C# example shows a call to the `Save` method.

```
Properties.Settings.Default.Save();
```

For general information about accessing application settings through the `Settings` class, see [Application settings overview \(.NET Framework\)](#). For information about iterating through the settings, see this [forum post](#).

See also

- [Access application settings \(.NET Framework\)](#)

How to: Add an application configuration file to a C# project

10/18/2019 • 2 minutes to read • [Edit Online](#)

By adding an application configuration file (*app.config* file) to a C# project, you can customize how the common language runtime locates and loads assembly files. For more information about application configuration files, see [How the runtime locates assemblies \(.NET Framework\)](#).

NOTE

UWP apps don't contain an *app.config* file.

When you build your project, the development environment automatically copies your *app.config* file, changes the file name of the copy to match your executable, and then moves the copy to the **bin** directory.

To add an application configuration file to a C# project

1. On the menu bar, choose **Project > Add New Item**.

The **Add New Item** dialog box appears.

2. Expand **Installed > Visual C# Items**, and then choose the **Application Configuration File** template.
3. In the **Name** text box, enter a name, and then choose the **Add** button.

A file named *app.config* is added to your project.

See also

- [Manage application settings \(.NET\)](#)
- [Configuration file schema \(.NET Framework\)](#)
- [Configure apps \(.NET Framework\)](#)

Manage assembly and manifest signing

10/21/2019 • 2 minutes to read • [Edit Online](#)

Strong-name signing gives a software component a globally unique identity. Strong names are used to guarantee that the assembly cannot be spoofed by someone else, and to make sure that component dependencies and configuration statements map to the correct component and component version.

A strong name consists of the assembly's identity (simple text name, version number, and culture information), plus a public key token and a digital signature.

For information about signing assemblies in Visual Basic and C# projects, see [Create and use strong-named assemblies](#).

For information about signing assemblies in C++ projects, see [Strong-named assemblies \(C++/CLI\)](#).

NOTE

Strong-name signing does not protect against reverse-engineering of the assembly. To protect against reverse-engineering, see [Dotfuscator Community](#).

Asset types and signing

You can sign .NET assemblies and application manifests:

- Executables (.exe)
- Application manifests (.exe.manifest)
- Deployment manifests (.application)
- Shared component assemblies (.dll)

Sign the following types of asset:

1. Assemblies, if you want to deploy them to the global assembly cache (GAC).
2. ClickOnce application and deployment manifests. Visual Studio enables signing by default for these applications.
3. Primary interop assemblies, which are used for COM interoperability. The TLBIMP utility enforces strong-naming when creating a primary interop assembly from a COM type library.

In general, you should not sign executables. A strongly named component cannot reference a non-strongly-named component that is deployed with the application. Visual Studio does not sign application executables, but instead signs the application manifest, which points to the weak-named executable. Avoid signing components that are private to your application, because signing can make it more difficult to manage dependencies.

How to sign an assembly in Visual Studio

You sign an application or component by using the **Signing** tab of the project properties window (right-click the project node in **Solution Explorer** and select **Properties**). Select the **Signing** tab, then select the **Sign the assembly** check box.

Specify a key file. If you choose to create a new key file, new key files are always created in the .pfx format. You

need a name and password for the new file.

WARNING

You should always protect your key file with a password to prevent someone else from using it. You can also secure your keys by using providers or certificate stores.

You can also point to a key you have already created. For more information about creating keys, see [Create a public-private key pair](#).

If you only have access to a public key, you can use delay-signing to defer assigning the key. You enable delay signing by selecting the **Delay sign only** check box. A delay-signed project doesn't run, and you can't debug it. However, you can skip verification during development by using the [Sn.exe strong name tool](#) with the `-vr` option.

For information about signing manifests, see [How to: Sign application and deployment manifests](#).

See also

- [Strong-named assemblies](#)
- [Strong-named assemblies \(C++/CLI\)](#)

How to: Sign application and deployment manifests

9/17/2019 • 4 minutes to read • [Edit Online](#)

If you want to publish an application by using ClickOnce deployment, the application and deployment manifests must be signed with a public/private key pair and signed using Authenticode technology. You can sign the manifests by using a certificate from the Windows certificate store or a key file.

For more information about ClickOnce deployment, see [ClickOnce security and deployment](#).

Signing the ClickOnce manifests is optional for .exe-based applications. For more information, see the "Generate unsigned manifests" section of this document.

For information about creating key files, see [How to: Create a public-private key pair](#).

NOTE

Visual Studio supports only Personal Information Exchange (PFX) key files that have the .pfx extension. However, you can select other types of certificates from the current user's Windows certificate store by clicking **Select from Store** on the **Signing** page of project properties.

Sign using a certificate

1. Go to the project properties window (right-click the project node in **Solution Explorer** and select **Properties**). On the **Signing** tab, select the **Sign the ClickOnce manifests** check box.
2. Click the **Select from Store** button.

The **Select a Certificate** dialog box appears and displays the contents of the Windows certificate store.

TIP

If you click **Click here to view certificate properties**, the **Certificate Details** dialog box appears. This dialog box includes detailed information about the certificate and additional options. Click **Certificates** to view additional help information.

3. Select the certificate that you want to use to sign the manifests.
4. Additionally, you can specify the address of a timestamp server in the **Timestamp server URL** text box. This is a server that provides a timestamp specifying when the manifest was signed.

Sign using an existing key file

1. On the **Signing** page, select the **Sign the ClickOnce manifests** check box.
2. Click the **Select from File** button.

The **Select File** dialog box appears.

3. In the **Select File** dialog box, browse to the location of the key file (.pfx) that you want to use, and then click **Open**.

NOTE

This option supports only files that have the .pfx extension. If you have a key file or certificate in another format, store it in the Windows certificate store and select the certificate is described in the previous procedure. The selected certificate's purpose should include code signing.

The **Enter password to open file** dialog box appears. (If the .pfx file is already stored in your Windows certificate store or is not password protected, you aren't prompted to enter a password.)

4. Enter the password to access the key file, and then select **Enter**.

NOTE

The .pfx file cannot include certificate chaining information. If it does, the following import error will occur: **Cannot find the certificate and private key for decryption**. To remove the certificate chaining information, you can use *Certmgr.msc* and disable the option to **Include all certificates** when exporting the *.pfx file.

Sign using a test certificate

1. On the **Signing** page, select the **Sign the ClickOnce manifests** check box.
2. To create a new certificate for testing, click the **Create Test Certificate** button.
3. In the **Create Test Certificate** dialog box, enter a password to help secure your test certificate.

Generate unsigned manifests

Signing the ClickOnce manifests is optional for .exe-based applications. The following procedures show how to generate unsigned ClickOnce manifests.

IMPORTANT

Unsigned manifests can simplify development and testing of your application. However, unsigned manifests introduce substantial security risks in a production environment. Only consider using unsigned manifests if your ClickOnce application runs on computers within an intranet that is completely isolated from the internet or other sources of malicious code.

By default, ClickOnce automatically generates signed manifests unless one or more files are specifically excluded from the generated hash. In other words, publishing the application results in signed manifests if all files are included in the hash, even when the **Sign the ClickOnce manifests** check box is cleared.

To generate unsigned manifests and include all files in the generated hash

1. To generate unsigned manifests that include all files in the hash, you must first publish the application together with signed manifests. Therefore, first sign the ClickOnce manifests by following one of the previous procedures, and then publish the application.
2. On the **Signing** page, clear the **Sign the ClickOnce manifests** check box.
3. Reset the publish version so that only one version of your application is available. By default, Visual Studio automatically increments the revision number of the publish version every time that you publish an application. For more information, see [How to: Set the ClickOnce publish version](#).
4. Publish the application.

To generate unsigned manifests and exclude one or more files from the generated hash

1. On the **Signing** page, clear the **Sign the ClickOnce manifests** check box.

2. Open the **Application Files** dialog box and set the **Hash to Exclude** for the files that you want to exclude from the generated hash.

NOTE

Excluding a file from the hash configures ClickOnce to disable automatic signing of the manifests, so you do not need to first publish with signed manifests as shown in the previous procedure.

3. Publish the application.

See also

- [Strong-named assemblies](#)
- [How to: Create a public-private key pair](#)
- [Signing page, Project Designer](#)
- [ClickOnce security and deployment](#)

How to: Specify an application icon (Visual Basic, C#)

10/18/2019 • 2 minutes to read • [Edit Online](#)

The **Icon** property for a project specifies the icon file (.ico) that will be displayed for the compiled application in **File Explorer** and in the Windows taskbar.

The **Icon** property can be accessed in the **Application** pane of the **Project Designer**; it contains a list of icons that have been added to a project either as resources or as content files.

NOTE

After you set the icon property for an application, you might also set the **Icon** property of each **Window** or **Form** in the application. For information about window icons for Windows Presentation Foundation (WPF) standalone applications, see [Icon property](#).

To specify an application icon

1. In **Solution Explorer**, choose a project node (not the **Solution** node).
2. On the menu bar, choose **Project > Properties**.
3. When the **Project Designer** appears, choose the **Application** tab.
4. **(Visual Basic)**—In the **Icon** list, choose an icon (.ico) file.
C#—Near the **Icon** list, choose the **<Browse...>** button, and then browse to the location of the icon file that you want.

See also

- [Application page, Project Designer \(Visual Basic\)](#)
- [Application page, Project Designer \(C#\)](#)

Framework targeting overview

10/18/2019 • 4 minutes to read • [Edit Online](#)

In Visual Studio, you can specify the version of .NET that you want your project to target. Framework targeting helps guarantee that the application uses only functionality that is available in the specified framework version. For .NET Framework apps to run on another computer, the framework version that the application targets must be compatible with the framework version that's installed on the computer.

A Visual Studio solution can contain projects that target different versions of .NET.

For more information about target frameworks, see [Target frameworks](#).

TIP

You can also target applications for different platforms. For more information, see [Multitargeting](#).

Framework targeting features

Framework targeting includes the following features:

- When you open a project that targets an earlier framework version, Visual Studio can automatically upgrade the project or leave the target as-is.
- When you create a .NET Framework project, you can specify the version of the .NET Framework that you want to target.
- You can [target multiple frameworks](#) in a single project.
- You can target a different version of .NET in each of several projects in the same solution.
- You can change the version of .NET that an existing project targets.

When you change the version of .NET that a project targets, Visual Studio makes any required changes to references and configuration files.

When you work on a project that targets an earlier framework version, Visual Studio dynamically changes the development environment, as follows:

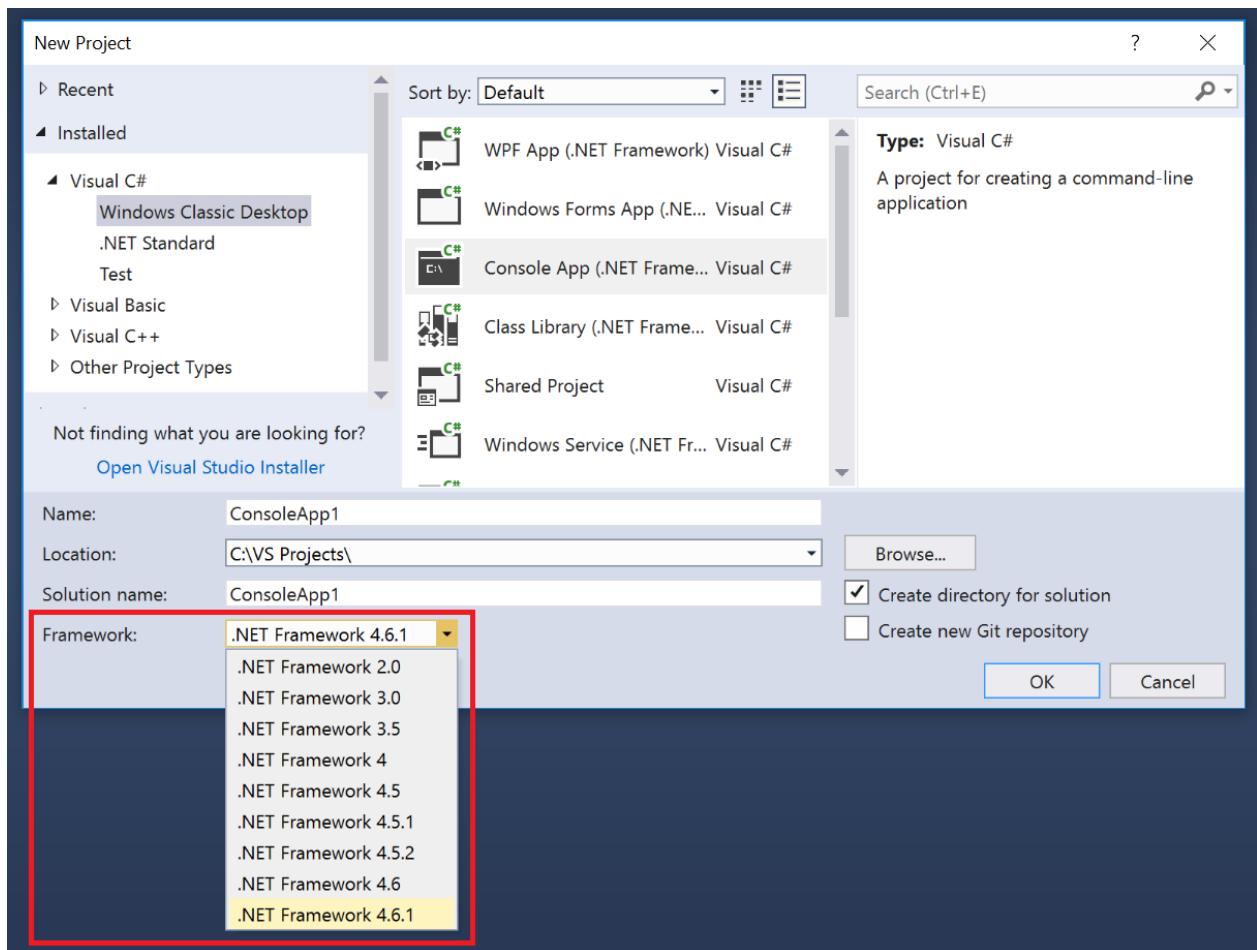
- It filters items in the **Add New Item** dialog box, the **Add New Reference** dialog box, and the **Add Service Reference** dialog box to omit choices that are not available in the targeted version.
- It filters custom controls in the **Toolbox** to remove those that are not available in the targeted version and to show the only the most up-to-date controls when multiple controls are available.
- It filters **IntelliSense** to omit language features that aren't available in the targeted version.
- It filters properties in the **Properties** window to omit those that aren't available in the targeted version.
- It filters menu options to omit options that aren't available in the targeted version.
- For builds, it uses the version of the compiler and the compiler options that are appropriate for the targeted version.

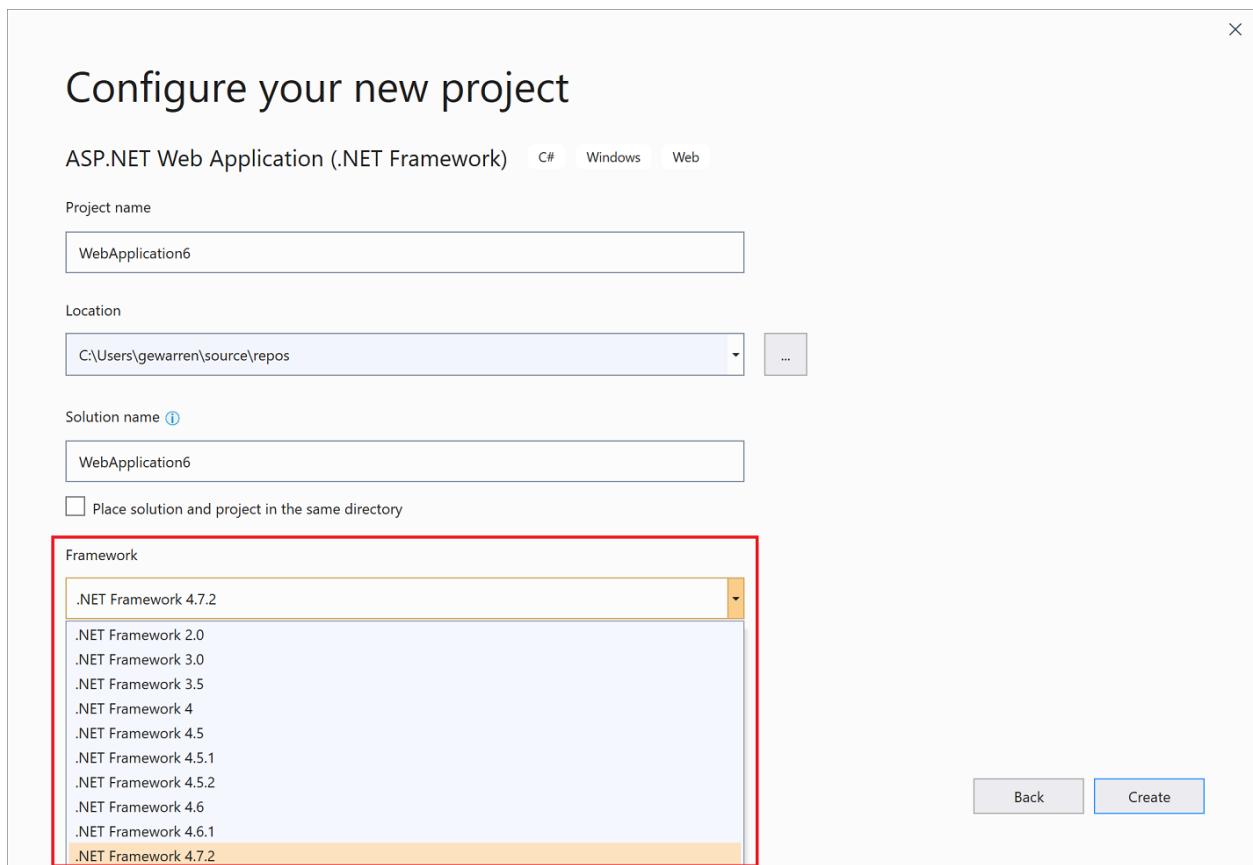
NOTE

- Framework targeting does not guarantee that your application will run correctly. You must test your application to make sure it runs against the targeted version.
- You cannot target framework versions below .NET Framework 2.0.

Select a target framework version

When you create a .NET Framework project, you can select the target .NET Framework version after you select a project template. The list of available frameworks includes the installed framework versions that are applicable to the selected template type. For non-.NET Framework project templates, for example .NET Core templates, the **Framework** drop-down list doesn't appear.

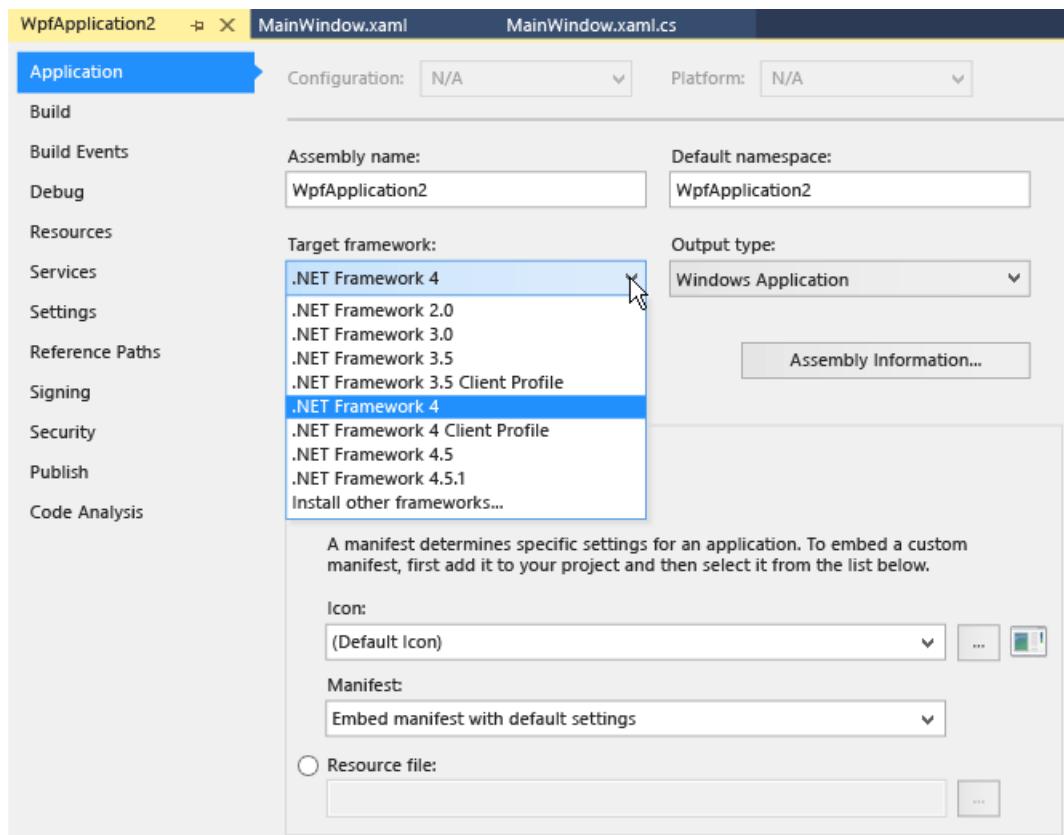




Change the target framework

In an existing Visual Basic, C#, or F# project, you change the target .NET version in the project properties dialog box. For information about how to change the target version for C++ projects, see [How to modify the target framework and platform toolset](#) instead.

1. In **Solution Explorer**, open the right-click menu for the project that you want to change, and then choose **Properties**.
2. In the left column of the **Properties** window, choose the **Application** tab.



NOTE

After you create a UWP app, you can't change the targeted version of either Windows or .NET.

3. In the **Target Framework** list, choose the version that you want.

4. In the verification dialog box that appears, choose the **Yes** button.

The project unloads. When it reloads, it targets the .NET version that you just chose.

NOTE

If your code contains references to a different version of the .NET than the one that you targeted, error messages may appear when you compile or run the code. To resolve these errors, modify the references. See [Troubleshoot .NET targeting errors](#).

TIP

Depending on the target framework, it can be represented in the following ways in the project file:

- For a .NET Core app: `<TargetFramework>netcoreapp2.1</TargetFramework>`
- For a .NET Standard app: `<TargetFramework>netstandard2.0</TargetFramework>`
- For a .NET Framework app: `<TargetFrameworkVersion>v4.7.2</TargetFrameworkVersion>`

Resolve system and user assembly references

To target a .NET version, you must first install the appropriate assembly references. You can download developer packs for different versions of .NET on the [.NET downloads](#) page.

For .NET Framework projects, the **Add Reference** dialog box disables system assemblies that do not pertain to the target .NET Framework version so that they cannot be inadvertently added to a project. (System assemblies

are *.dll* files that are included in a .NET Framework version.) References that belong to a framework version that's higher than the targeted version will not resolve, and controls that depend on such a reference cannot be added. If you want to enable such a reference, reset the .NET Framework target of the project to one that includes the reference.

For more information about assembly references, see [Resolve assemblies at design time](#).

Enable LINQ

When you target the .NET Framework 3.5 or later, a reference to **System.Core** and a project-level import for **System.Linq** (in Visual Basic only) are added automatically. If you want to use LINQ features, you must also turn `Option Infer` on (in Visual Basic only). The reference and import are removed automatically if you change the target to an earlier .NET Framework version. For more information, see [Work with LINQ](#).

See also

- [Target frameworks](#)
- [Multitargeting \(MSBuild\)](#)
- [How to: Modify the target framework and platform toolset \(C++\)](#)

Project and item templates

10/18/2019 • 2 minutes to read • [Edit Online](#)

Project and item templates provide reusable stubs that give users some basic code and structure, that they can customize for their own purposes.

Visual Studio templates

A number of predefined project and item templates are installed with Visual Studio. These templates, such as the **ASP.NET Web Application** and **Class Library** templates, are available to choose from when you create a new project. Item templates, such as code files, XML files, HTML pages, and Style Sheets, appear in the **Add New Item** window.

These templates provide a starting point for users to begin creating projects, or to expand existing projects. Project templates provide the files that are required for a particular project type, include standard assembly references, and set default project properties and compiler options. Item templates can range in complexity from a single empty file that has a certain file extension, to multiple source code files with stub code, designer information files, and embedded resources.

You can use installed templates, author your own custom templates, or download and use templates created by the community. For more information, see [How to: Create project templates](#) and [How to: Create item templates](#).

Contents of a template

All project and item templates, whether installed with Visual Studio or created by you, function using the same principles and have similar contents. All templates contain the following items:

- The files to be created when the template is used. These files include source code files, embedded resources, project files, and so on.
- A *.vstemplate* file, which contains the metadata needed to create a project or item from the template and to display the template in the **New Project** and **Add New Item** windows.
- A *.vstemplate* file, which contains the metadata needed to create a project or item from the template and to display the template on the **Create a new project** page or in the **Add New Item** dialog box.

For more information about *.vstemplate* files, see [Template tags](#) and [Template parameters](#).

When these files are compressed into a *.zip* file and put in the correct folder, Visual Studio automatically displays them in the following places:

- Project templates appear in the **New Project** window.
- Project templates appear on the **Create a new project** page.
- Item templates appear in the **Add New Item** window.

For more information about template folders, see [How to: Locate and organize templates](#).

See also

- [How to: Create project templates](#)
- [How to: Create item templates](#)

- [Template tags](#)
- [Template parameters](#)
- [Customize templates](#)
- [NuGet packages in Visual Studio templates](#)

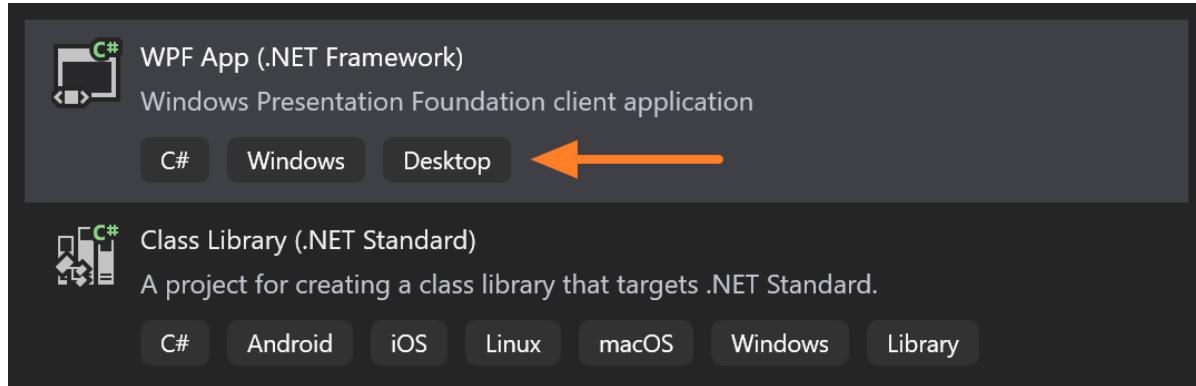
Add tags to project templates

10/31/2019 • 2 minutes to read • [Edit Online](#)

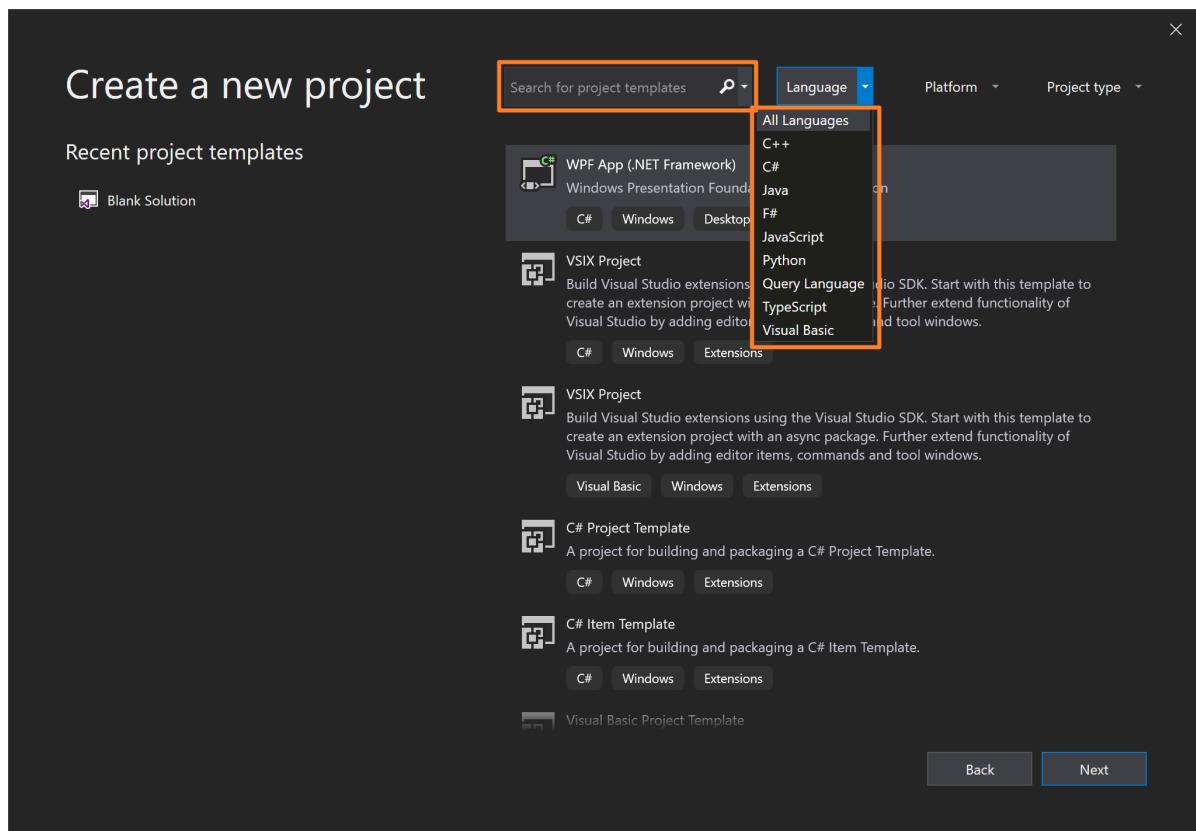
Starting in [Visual Studio 2019](#) version 16.1 Preview 2, you can add language, platform, and project type tags to your project templates.

Tags are used in two places in the **New Project** dialog box:

- Tags appear under the template description.



- Tags enable the template to be searched and filtered.



You can add tags by updating the `.vstemplate` XML file. You can either use template tags that are built into Visual Studio or create custom template tags. Template tags appear only in the Visual Studio 2019 **New Project** dialog box. Template tags don't affect how the template renders in earlier versions of Visual Studio.

Add or edit tags

You might want to add or edit tags in your project template's `.vstemplate` XML when you take one of the following actions:

- [Create a new project template](#) by using the Export Template wizard.
- [Update your existing project template](#).
- [Create a new VSIX project template](#).

Syntax

```
<LanguageTag> Language Name </LanguageTag>
<PlatformTag> Platform Name </PlatformTag>
<ProjectTypeTag> Project Type </ProjectTypeTag>
```

Attributes

You can use the following optional attributes in advanced user scenarios:

ATTRIBUTE	DESCRIPTION
<code>Package</code>	A GUID that specifies the Visual Studio package ID.
<code>ID</code>	Specifies the Visual Studio resource ID.

Syntax:

```
<LanguageTag Package="{PackageID}" ID="ResourceID" />
<PlatformTag Package="{PackageID}" ID="ResourceID" />
<ProjectTypeTag Package="{PackageID}" ID="ResourceID" />
```

Elements

Child elements

None.

Parent elements

ELEMENT	DESCRIPTION
<code>TemplateData</code>	(Required) Categorizes the template and defines how it displays in either the New Project dialog box or the Add New Item dialog box.

Text value

A text value is required unless you use the `Package` and `ID` attributes.

The text provides the name of the template.

Built-in tags

Visual Studio offers a list of built-in tags. When you add a built-in tag, the tag renders a localized resource.

The following list shows built-in tags that are available in Visual Studio. Corresponding values are shown in

parentheses.

LANGUAGE	PLATFORM	PROJECT TYPE
C++ (<code>cpp</code>)	Android (<code>android</code>)	Cloud (<code>cloud</code>)
C# (<code>csharp</code>)	Azure (<code>azure</code>)	Console (<code>console</code>)
F# (<code>fsharp</code>)	iOS (<code>ios</code>)	Desktop (<code>desktop</code>)
Java (<code>java</code>)	Linux (<code>linux</code>)	Extensions (<code>extension</code>)
JavaScript (<code>javascript</code>)	macOS (<code>macos</code>)	Games (<code>games</code>)
Python (<code>python</code>)	tvOS (<code>tvos</code>)	IoT (<code>iot</code>)
Query Languate (<code>querylanguage</code>)	Windows (<code>windows</code>)	Library (<code>library</code>)
TypeScript (<code>typescript</code>)	Xbox (<code>xbox</code>)	Machine Learning (<code>machinelearning</code>)
Visual Basic (<code>visualbasic</code>)		Mobile (<code>mobile</code>)
		Office (<code>office</code>)
		Other (<code>other</code>)
		Service (<code>service</code>)
		Test (<code>test</code>)
		UWP (<code>uwp</code>)
		Web (<code>web</code>)

Example

The following example shows the metadata for a project template for a Visual C# application:

```
<VSTemplate Type="Project" Version="3.0.0"
  xmlns="http://schemas.microsoft.com/developer/vstemplate/2005">
  <TemplateData>
    <Name>My template</Name>
    <Description>A basic template</Description>
    <Icon>TemplateIcon.ico</Icon>
    <ProjectType>CSharp</ProjectType>
    <LanguageTag>C#</LanguageTag>
    <PlatformTag>windows</PlatformTag>
    <PlatformTag>linux</PlatformTag>
    <PlatformTag>My Platform</PlatformTag>
    <ProjectTypeTag>console</ProjectTypeTag>
    <ProjectTypeTag>desktop</ProjectTypeTag>
  </TemplateData>
  <TemplateContent>
    <Project File="MyTemplate.csproj">
      <ProjectItem>Form1.cs</ProjectItem>
      <ProjectItem>Form1.Designer.cs</ProjectItem>
      <ProjectItem>Program.cs</ProjectItem>
      <ProjectItem>Properties\AssemblyInfo.cs</ProjectItem>
      <ProjectItem>Properties\Resources.resx</ProjectItem>
      <ProjectItem>Properties\Resources.Designer.cs</ProjectItem>
      <ProjectItem>Properties\Settings.settings</ProjectItem>
      <ProjectItem>Properties\Settings.Designer.cs</ProjectItem>
    </Project>
  </TemplateContent>
</VSTemplate>
```

See also

- [Visual Studio template schema reference](#)
- [Create project and item templates](#)
- [Customize project and item templates](#)
- [Get started with the VSIX project template](#)

How to: Create project templates

10/18/2019 • 2 minutes to read • [Edit Online](#)

This topic shows you how to create a template using the **Export Template Wizard**, which packages your template in a *.zip* file.

Use the Export Template Wizard

1. Create a project.

NOTE

Use only valid identifier characters when naming a project that will be the source for a template. Otherwise, compilation errors can occur in projects that are created from the template. For more information about valid identifier characters, see [Declared element names \(Visual Basic\)](#) or [Identifiers \(C++\)](#). Alternatively, you can use [Template parameters](#) to use "safe" names for classes and namespaces.

2. Edit the project until it is ready to be exported as a template. For example, you might want to edit code files to indicate where parameter replacement should take place. See [How to: Substitute parameters in a template](#).
3. On the **Project** menu, choose **Export Template**.

The **Export Template Wizard** opens.

4. On the **Choose Template Type** page, select **Project Template**. Select the project you want to export to a template, and then choose **Next**.
5. On the **Select Template Options** page, enter a name and optional description, icon, and preview image for your template. These items will appear in the **New Project** dialog box. Choose **Finish**.

The project is exported into a *.zip* file and placed in the specified output location, and, if selected, imported into Visual Studio.

To find your template in the **New Project** dialog box, expand **Installed** and then expand the category that corresponds to the `ProjectType` element in the *.vstemplate* file. For example, a *.vstemplate* file that contains `<ProjectType>CSharp</ProjectType>` appears under **Installed > Visual C#**, by default. You can organize your template into a subdirectory of the project type just by creating a folder in that directory and placing your template's *.zip* file in it. For more information, see [How to: Locate and organize templates](#).

5. On the **Select Template Options** page, enter a name and optional description, icon, and preview image for your template. These items will appear in the dialog box where you create a new project. Choose **Finish**.

The project is exported into a *.zip* file and placed in the specified output location, and, if selected, imported into Visual Studio.

To find your template in the dialog box where you create a new project, search for it by name or scroll through the list. (Filtering based on language or project type is not currently possible for user templates.)

Other ways to create project templates

You can create project templates manually by gathering the files that constitute the project into a folder and

creating a `.vstemplate` XML file with the appropriate metadata. For more information, see [How to: Manually create web templates](#).

If you have the Visual Studio SDK installed, you can wrap the finished template in a VSIX file for deployment by using the **VSIX Project** template. For more information, see [Get started with the VSIX project template](#).

See also

- [Create project and item templates](#)
- [How to: Create item templates](#)
- [Get started with the VSIX project template](#)

How to: Create multi-project templates

10/18/2019 • 4 minutes to read • [Edit Online](#)

Multi-project templates act as containers for two or more projects. When you create a project that's based on a multi-project template, every project in the template is added to the solution.

A multi-project template has two or more project templates and a root template of type **ProjectGroup**.

Multi-project templates behave differently than single project templates. They have the following unique characteristics:

- Individual projects in a multi-project template cannot be assigned names when the template is used to create a new project. Instead, use the **ProjectName** attribute on the **ProjectTemplateLink** element in the *vstemplate* file to specify a name for each project.
- Multi-project templates can contain projects for different languages, but the entire template itself can only be put in one category. Specify the template category in the **ProjectType** element of the *vstemplate* file.

A multi-project template must include the following items, compressed into a *.zip* file:

- A root *vstemplate* file for the entire multi-project template. This root *vstemplate* file contains metadata that's displayed in the dialog box where you create a new project. It also specifies where to find the *vstemplate* files for the projects in the template. This file must be located at the root of the *.zip* file.
- Two or more folders that contain the files that are required for a complete project template. The folders include all code files for the project, and also a *vstemplate* file for the project.

For example, a multi-project template *.zip* file that has two projects could have the following files and directories:

- *MultiProjectTemplate.vstemplate*
- *\Project1\MyTemplate.vstemplate*
- *\Project1\Project1.vbproj*
- *\Project1\Class.vb*
- *\Project2\MyTemplate.vstemplate*
- *\Project2\Project2.vbproj*
- *\Project2\Class.vb*

The root *vstemplate* file for a multi-project template differs from a single-project template in the following ways:

- The **Type** attribute of the **VSTemplate** element has the value **ProjectGroup** instead of **Project**. For example:

```
<VSTemplate Version="2.0.0" Type="ProjectGroup"
    xmlns="http://schemas.microsoft.com/developer/vstemplate/2005">
```

- The **TemplateContent** element contains a **ProjectCollection** element that has one or more **ProjectTemplateLink** elements that define the paths to the *vstemplate* files of the included projects. For example:

```
<TemplateContent>
  <ProjectCollection>
    <ProjectTemplateLink>
      Project1\MyTemplate.vstemplate
    </ProjectTemplateLink>
    <ProjectTemplateLink>
      Project2\MyTemplate.vstemplate
    </ProjectTemplateLink>
  </ProjectCollection>
</TemplateContent>
```

TIP

If you only want the multi-project template to appear in the new project dialog box and not the individual projects it contains, mark the inner templates as **hidden**. For example:

```
<VSTemplate Type="Project" ... >
  <TemplateData>
    ...
    <Hidden>true</Hidden>
  </TemplateData>
  ...
</VSTemplate>
```

Create a multi-project template from an existing solution

1. Create a solution and add two or more projects.
2. Customize the projects until they are ready to be exported to a template.

TIP

If you're using **template parameters** and you want to refer to variables from the parent template, prefix the name of the parameter with `ext_`. For example, `$ext_safeprojectname$`. Also, set the **CopyParameters** attribute of the **ProjectTemplateLink** element to **true**.

```
<ProjectTemplateLink ProjectName="MyProject" CopyParameters="true">...</ProjectTemplateLink>
```

3. On the **Project** menu, choose **Export Template**.

The **Export Template Wizard** opens.

4. On the **Choose Template Type** page, select **Project Template**. Select one of the projects that you want to export to a template, and then choose **Next**. (You'll repeat these steps for each project in the solution.)
5. On the **Select Template Options** page, enter a name and optional description, icon, and preview image for your template. Choose **Finish**.

The project is exported into a `.zip` file and placed in the specified output location.

NOTE

Each project must be exported to a template separately, so repeat the preceding steps for each project in the solution.

6. Create a directory for your template, with a subdirectory for each project.

7. Extract the contents of each project's *.zip* file into the corresponding subdirectory that you created.
 8. In the base directory, create an XML file with a *.vstemplate* file extension. This file contains the metadata for the multi-project template. See the example that follows for the structure of the file. Be sure to specify the relative path to each project's *vstemplate* file.
 9. Select all the files in the base directory, and from the right-click or context menu, choose **Send to > Compressed (zipped) folder**.
- The files and folders are compressed into a *.zip* file.
10. Copy the *.zip* file into the user project template directory. By default, this directory is `%USERPROFILE%\Documents\Visual Studio <version>\Templates\ProjectTemplates`.
 11. In Visual Studio, choose **File > New > Project** and verify that your template appears.

Two-project example

This example shows a basic multi-project root *vstemplate* file. In this example, the template has two projects, **My Windows Application** and **My Class Library**. The **ProjectName** attribute on the **ProjectTemplateLink** element specifies the name that is given to the project.

TIP

If the **ProjectName** attribute is not specified, the name of the *vstemplate* file is used as the project name.

```
<VSTemplate Version="2.0.0" Type="ProjectGroup"
  xmlns="http://schemas.microsoft.com/developer/vstemplate/2005">
  <TemplateData>
    <Name>Multi-Project Template Sample</Name>
    <Description>An example of a multi-project template</Description>
    <Icon>Icon.ico</Icon>
    <ProjectType>VisualBasic</ProjectType>
  </TemplateData>
  <TemplateContent>
    <ProjectCollection>
      <ProjectTemplateLink ProjectName="My Windows Application">
        WindowsApp\MyTemplate.vstemplate
      </ProjectTemplateLink>
      <ProjectTemplateLink ProjectName="My Class Library">
        ClassLib\MyTemplate.vstemplate
      </ProjectTemplateLink>
    </ProjectCollection>
  </TemplateContent>
</VSTemplate>
```

Example with solution folders

This example uses the **SolutionFolder** element to divide the projects into two groups, **Math Classes** and **Graphics Classes**. The template has four projects, two of which are placed in each solution folder.

```
<VSTemplate Version="2.0.0" Type="ProjectGroup"
  xmlns="http://schemas.microsoft.com/developer/vstemplate/2005">
  <TemplateData>
    <Name>Multi-Project Template Sample</Name>
    <Description>An example of a multi-project template</Description>
    <Icon>Icon.ico</Icon>
    <ProjectType>VisualBasic</ProjectType>
  </TemplateData>
  <TemplateContent>
    <ProjectCollection>
      <SolutionFolder Name="Math Classes">
        <ProjectTemplateLink ProjectName="MathClassLib1">
          MathClassLib1\MyTemplate.vstemplate
        </ProjectTemplateLink>
        <ProjectTemplateLink ProjectName="MathClassLib2">
          MathClassLib2\MyTemplate.vstemplate
        </ProjectTemplateLink>
      </SolutionFolder>
      <SolutionFolder Name="Graphics Classes">
        <ProjectTemplateLink ProjectName="GraphicsClassLib1">
          GraphicsClassLib1\MyTemplate.vstemplate
        </ProjectTemplateLink>
        <ProjectTemplateLink ProjectName="GraphicsClassLib2">
          GraphicsClassLib2\MyTemplate.vstemplate
        </ProjectTemplateLink>
      </SolutionFolder>
    </ProjectCollection>
  </TemplateContent>
</VSTemplate>
```

See also

- [Creating project and item templates](#)
- [How to: Create project templates](#)
- [Visual Studio template schema reference \(extensibility\)](#)
- [SolutionFolder element \(Visual Studio templates\)](#)
- [ProjectTemplateLink element \(Visual Studio templates\)](#)

How to: Create item templates

10/18/2019 • 4 minutes to read • [Edit Online](#)

This article shows you how to create an item template by using the **Export Template Wizard**. If your template will consist of multiple files, see [How to: Create multi-file item templates](#).

Add an item template to the Add New Item dialog box

1. Create or open a project in Visual Studio.
2. Add an item to the project, and modify it if you want to.
3. Modify the code file to indicate where parameter replacement should take place. For more information, see [How to: Substitute parameters in a template](#).
4. On the **Project** menu, choose **Export Template**.
5. On the **Choose Template Type** page, choose **Item Template**, select the project that contains the item, and then choose **Next**.
6. On the **Select Item To Export** page, choose the item you want to create a template for, and then choose **Next**.
7. On the **Select Item References** page, select the assembly references to include in the template, and then choose **Next**.
8. On the **Select Template Options** page, enter the template name and optional description, icon image and preview image, and then choose **Finish**.

The files for the template are added to a .zip file and copied to the directory you specified in the wizard. The default location is %USERPROFILE%\Documents\Visual Studio <version>\My Exported Templates.

9. If you did not select the option **Automatically import the template into Visual Studio** in the **Export Template Wizard**, locate the exported template. Then, copy it to the user item template directory. The default location is %USERPROFILE%\Documents\Visual Studio <version>\Templates\ItemTemplates.
10. Close Visual Studio and then reopen it.
11. Create a new project, or open an existing project, and then choose **Project > Add New Item** or press **Ctrl+Shift+A**.

The item template appears in the **Add New Item** dialog box. If you added a description in the **Export Template Wizard**, the description appears on the right side of the dialog box.

Enable the item template to be used in a Universal Windows App project

The wizard does much of the work to create a basic template, but in many cases you need to manually modify the .vstemplate file after you have exported the template. For example, if you want the item to appear in the **Add New Item** dialog for a Universal Windows App project, you have to perform a few extra steps.

1. Follow the steps in the previous section to export an item template.
2. Extract the .zip file that was created, and open the .vstemplate file in Visual Studio.

3. For a C# Universal Windows project, add the following XML inside the `<TemplateData>` element:

```
<TemplateID>Microsoft.CSharp.Class</TemplateID>
```

4. In Visual Studio, save the `.vstemplate` file and close it.

5. Copy and paste the `.vstemplate` file back to the `.zip` file.

If the **Copy File** dialog box appears, choose the **Copy and Replace** option.

You can now add an item based on this template to a Universal Windows project from the **Add New Item** dialog box.

Enable templates for specific project subtypes

You can specify that your template should only appear for only certain project subtypes, such as Windows, Office, Database, or Web.

1. Locate the `ProjectType` element in the `.vstemplate` file for the item template.

2. Add a `ProjectSubType` element immediately after the `ProjectType` element.

3. Set the text value of the element to one of the following values:

- Windows
- Office
- Database
- Web

For example: `<ProjectSubType>Database</ProjectSubType>`.

The following example shows an item template for **Office** projects.

```
<VSTemplate Version="2.0.0" Type="Item" Version="2.0.0">
  <TemplateData>
    <Name>Class</Name>
    <Description>An empty class file</Description>
    <Icon>Class.ico</Icon>
    <ProjectType>CSharp</ProjectType>
    <ProjectSubType>Office</ProjectSubType>
    <DefaultName>Class.cs</DefaultName>
  </TemplateData>
  <TemplateContent>
    <ProjectItem>Class1.cs</ProjectItem>
  </TemplateContent>
</VSTemplate>
```

Manually create an item template

In some cases you may want to create an item template manually, from scratch.

1. Create a project and project item.
2. Modify the project item until it is ready to be saved as a template.
3. Modify the code file to indicate where parameter replacement should occur, if anywhere. For more information about parameter replacement, see [How to: Substitute parameters in a template](#).
4. Create an XML file and save it with a `.vstemplate` file extension in the same directory as your project item

file.

5. Edit the `.vstemplate` XML file to provide item template metadata. For more information, see [Template schema reference \(extensibility\)](#) and the example in the previous section.
6. Save the `.vstemplate` file and close it.
7. In **Windows Explorer**, select the files you want to include in your template. Right-click the selection, and choose **Send to > Compressed (zipped) folder**. The files that you selected are compressed into a `.zip` file.
8. Copy the `.zip` file and paste it in the user item template location. The default directory is `%USERPROFILE%\Documents\Visual Studio 2017\Templates\ItemTemplates`. For more information, see [How to: Locate and organize project and item templates](#).
8. Copy the `.zip` file and paste it in the user item template location. The default directory is `%USERPROFILE%\Documents\Visual Studio 2019\Templates\ItemTemplates`. For more information, see [How to: Locate and organize project and item templates](#).

See also

- [Create project and item templates](#)
- [How to: Create multi-file item templates](#)
- [Visual Studio template schema reference \(extensibility\)](#)

How to: Create multi-file item templates

10/18/2019 • 2 minutes to read • [Edit Online](#)

Item templates may only specify one item, but sometimes the item is made up of multiple files. For example, a Windows Forms item template requires the following three files:

- A file that contains the code for the form
- A file that contains the designer information for the form
- A file that contains the embedded resources for the form

Multi-file item templates require parameters to ensure that the correct file extensions are used when the item is created. If you create a multi-file item template by using the **Export Template Wizard**, these parameters are automatically generated, and no further editing is required.

Use the Export Template Wizard

You can create a multi-file item template in the same manner as you would a single-file item template. See [How to: Create item templates](#). On the **Select Item To Export** page of the wizard, select the file that has dependent files (for example, a Windows Forms form file). The wizard automatically includes any dependent files, such as designer and resource files, in the template.

Manually create a multi-file item template

1. Create the item template as you would manually create a single-file item template, but include each file that constitutes the multi-file item.
2. In the `.vstemplate` XML file, add a `<ProjectItem>` element for each individual file, and add a `TargetFileName` attribute to this element. Set the value of the `TargetFileName` attribute to `$fileinputname$.FileExtension`, where `FileExtension` is the file extension of the file that is being included in the template. For example:

```
<ProjectItem TargetFileName="$fileinputname$.vb">
  Form1.vb
</ProjectItem>
<ProjectItem TargetFileName="$fileinputname$.Designer.vb">
  Form1.Designer.vb
</ProjectItem>
<ProjectItem TargetFileName="$fileinputname$.resx">
  Form1.resx
</ProjectItem>
```

NOTE

When an item derived from this template is added to a project, the file names will derive from the name that the user enters in the **Add New Item** dialog box.

3. Select the files to be included in your template, right-click the selection, and choose **Send to > Compressed (zipped) folder**.

The files that you selected are compressed into a `.zip` file.

4. Copy the `.zip` file to the user item template location. By default, the directory is

`%USERPROFILE%\Documents\Visual Studio <Version>\Templates\ItemTemplates`. For more information, see [How to: Locate and organize templates](#).

5. Close Visual Studio and then reopen it.
6. Create a new project, or open an existing project, and then choose **Project > Add New Item** or press **Ctrl+Shift+A**.

The multi-file item template appears in the **Add New Item** dialog box.

Example

The following example shows a Windows Forms template. When an item is created based on this template, the names of the three files created will match the name entered in the **Add New Item** dialog box.

```
<VSTemplate Version="2.0.0" Type="Item"
  xmlns="http://schemas.microsoft.com/developer/vstemplate/2005">
  <TemplateData>
    <Name>Multi-file Item Template</Name>
    <Icon>Icon.ico</Icon>
    <Description>An example of a multi-file item template</Description>
    <ProjectType>VisualBasic</ProjectType>
  </TemplateData>
  <TemplateContent>
    <ProjectItem TargetFileName="$fileinputname$.vb" SubType="Form">
      Form1.vb
    </ProjectItem>
    <ProjectItem TargetFileName="$fileinputname$.Designer.vb">
      Form1.Designer.vb
    </ProjectItem>
    <ProjectItem TargetFileName="$fileinputname$.resx">
      Form1.resx
    </ProjectItem>
  </TemplateContent>
</VSTemplate>
```

See also

- [Create project and item templates](#)
- [How to: Create item templates](#)
- [Template parameters](#)
- [How to: Substitute parameters in a template](#)

How to: Manually create web templates

10/18/2019 • 2 minutes to read • [Edit Online](#)

Creating a web template is different than creating other kinds of templates. Because web project templates appear in the **Add New Web Site** dialog box, and web project items are categorized by programming language, the *vstemplate* file must specify the template as a web template and identify the programming language.

NOTE

Web templates must contain an empty *.webproj* file, and it must be referenced in the *vstemplate* file in the **File** attribute of the **Project** element. Although web projects do not require a *.proj* project file, it's necessary to create this stub file for the web template to function correctly.

To manually create a web template

1. Create a web project.
2. Modify or delete the files in the project, or add new files to the project.
3. Create an XML file and save it with a *vstemplate* file name extension, in the same directory as your project.
Do not add it to the project in Visual Studio.
4. Edit the *vstemplate* XML file to provide project template metadata. For more information, see the [example that follows](#).
5. Locate the **ProjectType** element in the *vstemplate* file, and set the text value to **Web**.
6. Following the **ProjectType** element, add a **ProjectSubType** element and set the text value to the programming language of the template. The programming language can be one of the following values:
 - CSharp
 - VisualBasic

For example:

```
<TemplateData>
  ...
  <ProjectType>Web</ProjectType>
  <ProjectSubType>CSharp</ProjectSubType>
  ...
</TemplateData>
```

7. Select the files in your template (this includes the *vstemplate* file), right-click the selection, and choose **Send to** > **Compressed (zipped) folder**. The files are compressed into a *.zip* file.
8. Put the *.zip* template file in the Visual Studio project template directory. By default, this directory is *%USERPROFILE%\Documents\Visual Studio <Version>\ProjectTemplates*.

Example

The following example shows a basic *vstemplate* file for a web project template:

```
<VSTemplate Version="2.0.0" Type="Project"
  xmlns="http://schemas.microsoft.com/developer/vstemplate/2005">
  <TemplateData>
    <Name>MyWebProjecStarterKit</Name>
    <Description>A simple web template</Description>
    <Icon>icon.ico</Icon>
    <ProjectType>Web</ProjectType>
    <ProjectSubType>CSharp</ProjectSubType>
    <DefaultName>WebSite</DefaultName>
  </TemplateData>
  <TemplateContent>
    <Project File="WebApplication.webproj">
      <ProjectItem>icon.ico</ProjectItem>
      <ProjectItem OpenInEditor="true">Default.aspx</ProjectItem>
      <ProjectItem>Default.aspx.cs</ProjectItem>
    </Project>
  </TemplateContent>
</VSTemplate>
```

See also

- [Create project and item templates](#)
- [Visual Studio template schema reference \(extensibility\)](#)

How to: Troubleshoot templates

10/18/2019 • 2 minutes to read • [Edit Online](#)

If a template fails to load in the development environment, there are several ways to locate the problem.

Validate the `vstemplate` file

If the `vstemplate` file in a template doesn't adhere to the Visual Studio template schema, the template may not appear in the **New Project** dialog box.

If the `vstemplate` file in a template doesn't adhere to the Visual Studio template schema, the template may not appear in the dialog box where you create new projects.

To validate the `vstemplate` file

1. Locate the `.zip` file that contains the template.
2. Extract the `.zip` file.
3. On the **File** menu in Visual Studio, choose **Open > File**.
4. Select the `vstemplate` file for the template, and choose **Open**.
5. Verify that the XML of the `vstemplate` file adheres to the template schema. For more information on the `vstemplate` schema, see [Template schema reference](#).

NOTE

To get IntelliSense support while authoring the `vstemplate` file, add a `xmlns` attribute to the `VSTemplate` element, and assign it a value of <http://schemas.microsoft.com/developer/vstemplate/2005>.

6. Save and close the `vstemplate` file.
7. Select the files included in your template, right-click, and choose **Send to > Compressed (zipped) folder**.
The files that you selected are compressed into a `.zip` file.
8. Place the new `.zip` file in the same directory as the old `.zip` file.
9. Delete the extracted template files and the old template `.zip` file.

Enable diagnostic logging

You can enable diagnostic logging for template discovery by following the steps in [Troubleshoot template discovery \(extensibility\)](#).

See also

- [Troubleshoot template discovery \(extensibility\)](#)
- [Customize templates](#)
- [Create project and item templates](#)
- [Template schema reference](#)

How to: Locate and organize project and item templates

10/18/2019 • 3 minutes to read • [Edit Online](#)

Template files must be placed in a known location in order for them to be shown in the new project and new item dialog boxes..

You can also create custom subcategories in the user template location, and the categories are shown in the **New Project** and **Add New Item** dialog boxes.

Locate templates

Installed templates and user templates are stored in two different locations.

Installed templates

By default, templates installed with Visual Studio are located in:

- *%ProgramFiles(x86)%\Microsoft Visual Studio\2017\<edition>\Common7\IDE\ProjectTemplates\<Language>\<Locale ID>*
- *%ProgramFiles(x86)%\Microsoft Visual Studio\2017\<edition>\Common7\IDE\ItemTemplates\<Language>\<Locale ID>*

For example, the following directory has the Visual Basic item templates for English (LCID 1033):

C:\Program Files (x86)\Microsoft Visual Studio\2017\Community\Common7\IDE\ItemTemplates\VisualBasic\1033

- *%ProgramFiles(x86)%\Microsoft Visual Studio\2019\<edition>\Common7\IDE\ProjectTemplates\<Language>\<Locale ID>*
- *%ProgramFiles(x86)%\Microsoft Visual Studio\2019\<edition>\Common7\IDE\ItemTemplates\<Language>\<Locale ID>*

For example, the following directory has the Visual Basic item templates for English (LCID 1033):

C:\Program Files (x86)\Microsoft Visual Studio\2019\Community\Common7\IDE\ItemTemplates\VisualBasic\1033

User templates

If you add a compressed (.zip) file that includes a *.vstemplate* file to the user template directory, the template appears in the new project and new item dialog boxes. By default, user templates are located in:

- *%USERPROFILE%\Documents\Visual Studio 2017\Templates\ProjectTemplates*
- *%USERPROFILE%\Documents\Visual Studio 2017\Templates\ItemTemplates*

For example, the following directory has user project templates for C#:

- *C:\Users\UserName\Documents\Visual Studio 2017\Templates\ProjectTemplates\Visual C#*
- *%USERPROFILE%\Documents\Visual Studio 2019\Templates\ProjectTemplates*
- *%USERPROFILE%\Documents\Visual Studio 2019\Templates\ItemTemplates*

For example, the following directory has user project templates for C#:

- C:\Users\UserName\Documents\Visual Studio 2019\Templates\ProjectTemplates\Visual C#

TIP

You can change the known location for user templates in **Tools > Options > Projects and Solutions > Locations**.

Organize templates

The categories in the **New Project** and **Add New Item** dialog boxes reflect the directory structures that exist in the installed template and user template locations. User templates can be organized into their own categories by adding new folders to the user template directory. The **New Project** and **Add New Item** dialog boxes show any changes you make to your user template categories.

NOTE

You cannot create a new category at the programming language level. New categories can only be created within each language.

Create new user project template categories

1. Create a folder in the programming language folder in the user project template directory. For example, to establish a **HelloWorld** category for C# project templates, create the following directory:
 - %USERPROFILE%\Documents\Visual Studio <Version>\Templates\ProjectTemplates\Visual C#\HelloWorld
2. Place all the templates for this category in the new folder.
3. On the **File** menu, choose **New > Project**.

The **HelloWorld** category appears in the **New Project** dialog box, under **Installed > Visual C#**.

Create new user item template categories

1. Create a folder in the programming language folder in the user item template directory. For example, to establish a **HelloWorld** category for C# item templates, create the following directory:
 - %USERPROFILE%\Documents\Visual Studio <Version>\Templates\ItemTemplates\Visual C#\HelloWorld
2. Place all the templates for this category in the new folder.
3. Create a project or open an existing project. Then, on the **Project** menu, choose **Add New Item**.

The **HelloWorld** category appears in the **Add New Item** dialog box, under **Installed > Visual C# Items**.

Display templates in parent categories

You can enable templates in subcategories to be displayed in their parent categories by using the `NumberOfParentCategoriesToRollUp` element in the `.vstemplate` file. These steps are the same for project templates and item templates.

1. Locate the `.zip` file that contains the template.
2. Extract the `.zip` file.
3. Open the `.vstemplate` file in Visual Studio.
4. In the `TemplateData` element, add a `NumberOfParentCategoriesToRollUp` element. For example, the following

code makes the template visible in the parent category, but no higher.

```
<TemplateData>
  ...
  <NumberOfParentCategoriesToRollUp>
    1
  </NumberOfParentCategoriesToRollUp>
  ...
</TemplateData>
```

5. Save and close the `.vstemplate` file.
6. Select the files in your template, right-click the selection, and choose **Send to > Compressed (zipped) folder**.
The files are compressed into a `.zip` file.
7. Delete the extracted template files and the old template `.zip` file.
8. Put the new `.zip` file in the directory that had the deleted `.zip` file.

See also

- [Customize templates](#)
- [Visual Studio template schema reference \(extensibility\)](#)
- [NumberOfParentCategoriesToRollUp \(Visual Studio templates\)](#)
- [How to: Create project templates](#)
- [How to: Create item templates](#)

Customize project and item templates

10/18/2019 • 2 minutes to read • [Edit Online](#)

Even after project and item templates have been created, you can further customize them to meet your needs.

Customizations

For example, you can perform the following tasks:

- Modify and export an existing template as a user template.

For more information, see [How to: Update existing templates](#).

- Pass custom parameters into a template to replace existing values.

For more information, see [How to: Substitute parameters in a template](#).

- Customize the wizards that create projects from templates.

For more information, see [How to: Use wizards with project templates \(extensibility\)](#).

See also

- [Creating project and item templates](#)
- [How to: Troubleshoot templates](#)
- [How to: Create project templates](#)
- [How to: Create item templates](#)
- [Visual Studio template schema reference](#)
- [IWizard](#)

How to: Update existing templates

10/18/2019 • 2 minutes to read • [Edit Online](#)

After you create a template and compress the files into a .zip file, you may want to modify the template. You can do this by manually changing the files in the template or by exporting a new template from a project that's based on the template.

Use the Export Template Wizard

Visual Studio provides an **Export Template Wizard** that can be used to update an existing template:

1. Choose **File > New > Project** from the menu bar.
2. Select the template that you want to update and continue through the steps to create the new project.
3. Modify the project in Visual Studio. For example, change the output type or add a new file to the project.
4. On the **Project** menu, choose **Export Template**.

The **Export Template Wizard** opens.

5. Follow the prompts in the wizard to export the template as a .zip file.
6. (Optional) Place the .zip file in the following directory: %USERPROFILE%\Documents\Visual Studio <version>\Templates\ProjectTemplates to make it available for selection. You'll need to perform this step if you did not select the option **Automatically import the template into Visual Studio** in the **Export Template Wizard**.
7. Delete the old template .zip file.

Manually update an existing template

You can update an existing template without using the **Export Template Wizard**, by modifying the files in the compressed .zip file.

To manually update an existing template

1. Locate the .zip file that contains the template. User project templates are located at %USERPROFILE%\Documents\Visual Studio <version>\Templates\ProjectTemplates.
2. Extract the .zip file.
3. Modify or delete the current template files, or add new files to the template.
4. Open, modify, and save the .vstemplate XML file to handle updated behavior or new files.

For more information about the .vstemplate schema, see [Visual Studio template schema reference \(extensibility\)](#). For more information about what you can parameterize in the source files, see [Template parameters](#).

5. Select the files in your template, and from the right-click or context menu, and choose **Send to > Compressed (zipped) folder**.

The files that you selected are compressed into a .zip file.

6. Put the new .zip file in the same directory as the old .zip file.

7. Delete the extracted template files and the old template *.zip* file.

See also

- [Customize templates](#)
- [Create project and item templates](#)
- [Visual Studio template schema reference](#)
- [Template parameters](#)

How to: Substitute parameters in a template

10/18/2019 • 2 minutes to read • [Edit Online](#)

Template parameters let you replace identifiers such as class names and namespaces when a file is created from a template. You can add template parameters to existing templates, or create your own templates with template parameters.

Template parameters are written in the format `$parameter$`. For a complete list of template parameters, see [Template parameters](#).

The following section shows you how to modify a template to replace the name of a namespace with the "safe project name".

Example - namespace name

1. Insert the parameter in one or more of the code files in the template. For example:

```
namespace $safeprojectname$
```

2. In the `vstemplate` file for the template, locate the `ProjectItem` element that includes this file.

3. Set the `ReplaceParameters` attribute to `true` for the `ProjectItem` element:

```
<ProjectItem ReplaceParameters="true">Class1.cs</ProjectItem>
```

See also

- [Create project and item templates](#)
- [Template parameters](#)
- [Visual Studio template schema reference](#)
- [ProjectItem element \(Visual Studio item templates\)](#)

Template parameters

10/18/2019 • 2 minutes to read • [Edit Online](#)

You can replace values in your template when the template is instantiated. To set up this functionality, use *template parameters*. Template parameters can be used to replace values such as class names and namespaces in the template. The template wizard that runs in the background when a user adds a new item or project replaces these parameters.

Declare and enable template parameters

Template parameters are declared in the format `$parameter$`. For example:

- `$safe projectName$`
- `$guid1$`
- `$guid5$`

Enable parameter substitution in templates

1. In the `.vstemplate` file of the template, locate the `ProjectItem` element that corresponds to the item for which you want to enable parameter replacement.
2. Set the `ReplaceParameters` attribute of the `ProjectItem` element to `true`.
3. In the code file for the project item, include parameters where appropriate. For example, the following parameter specifies that the safe project name is used for the namespace in a file:

```
namespace $safe projectName$
```

Reserved template parameters

The following table lists the reserved template parameters that can be used by any template:

PARAMETER	DESCRIPTION
<code>clrversion</code>	Current version of the common language runtime (CLR).
<code>ext_*</code>	Add the <code>ext_</code> prefix to any parameter to refer to the variables of the parent template. For example, <code>ext_safe projectName</code> .
<code>guid[1-10]</code>	A GUID used to replace the project GUID in a project file. You can specify up to 10 unique GUIDs (for example, <code>guid1</code>).
<code>itemname</code>	The name of the file in which the parameter is being used.
<code>machinename</code>	The current computer name (for example, Computer01).
<code>projectname</code>	The name provided by the user when the project was created.

PARAMETER	DESCRIPTION
registeredorganization	The registry key value from HKLM\Software\Microsoft\Windows NT\CurrentVersion\RegisteredOrganization.
rootnamespace	The root namespace of the current project. This parameter applies only to item templates.
safeitemname	Same as <code>itemname</code> but with all unsafe characters and spaces replaced by underscore characters.
safeitemrootname	Same as <code>safeitemname</code> .
safeprojectname	The name provided by the user when the project was created but with all unsafe characters and spaces removed.
time	The current time in the format DD/MM/YYYY 00:00:00.
specifiedSolutionName	The name of the solution. When "create solution directory" is checked, <code>specifiedSolutionName</code> has the solution name. When "create solution directory" is not checked, <code>specifiedSolutionName</code> is blank.
userdomain	The current user domain.
username	The current user name.
webnamespace	The name of the current website. This parameter is used in the web form template to guarantee unique class names. If the website is at the root directory of the web server, this template parameter resolves to the root directory of the web server.
year	The current year in the format YYYY.

NOTE

Template parameters are case-sensitive.

Custom template parameters

You can specify your own template parameters and values, in addition to the default reserved template parameters that are used during parameter replacement. For more information, see [CustomParameters element \(Visual Studio templates\)](#).

Example: Use the project name for a file name

You can specify variable file names for project items by using a parameter in the `TargetFileName` attribute.

The following example specifies that an executable file's name uses the project name, specified by `$projectname$`.

```
<TemplateContent>
  <ProjectItem
    ReplaceParameters="true"
    TargetFileName="$projectname$.exe">
    File1.exe
  </ProjectItem>
  ...
</TemplateContent>
```

Example: Use the safe project name for the namespace name

To use the safe project name for the namespace in a C# class file, use the following syntax:

```
namespace $safeprojectname$
{
  public class Class1
  {
    public Class1()
    { }
  }
}
```

In the *.vstemplate* file for the project template, include the `ReplaceParameters="true"` attribute when you reference the file:

```
<TemplateContent>
  <ProjectItem ReplaceParameters="true">
    Class1.cs
  </ProjectItem>
  ...
</TemplateContent>
```

See also

- [How to: Substitute parameters in a template](#)
- [Customize templates](#)
- [How to: Create project templates](#)
- [Template Schema Reference](#)

Visual Studio IDE 64-Bit support

10/18/2019 • 2 minutes to read • [Edit Online](#)

Visual Studio enables you to set up your applications to target different platforms, including 64-bit platforms. For more information on 64-bit platform support in Visual Studio, see [64-bit applications](#).

Deploy a 64-bit application

[Deploy prerequisites for 64-bit applications](#) lists the redistributables you can use as prerequisites for the installation of a 64-bit application.

Configure projects as 64-bit applications

[How to: Configure projects to target platforms](#) discusses configuring projects to be built as 64-bit applications.

Debug a 64-bit application

- [Debug 64-bit applications](#)
- [Use dump files](#)

Develop code in Visual Studio without projects or solutions

10/18/2019 • 4 minutes to read • [Edit Online](#)

You can open code from nearly any type of directory-based project into Visual Studio without the need for a solution or project file. This means you can, for example, clone a repo on GitHub, open it directly into Visual Studio, and begin developing, without having to create a solution or project. If needed, you can specify custom build tasks and launch parameters through simple JSON files.

After you open your code files in Visual Studio, **Solution Explorer** displays all the files in the folder. You can click on any file to begin editing it. In the background, Visual Studio starts indexing the files to enable IntelliSense, navigation, and refactoring features. As you edit, create, move, or delete files, Visual Studio tracks the changes automatically and continuously updates its IntelliSense index. Code will appear with syntax colorization and, in many cases, include basic IntelliSense statement completion.

Open any code

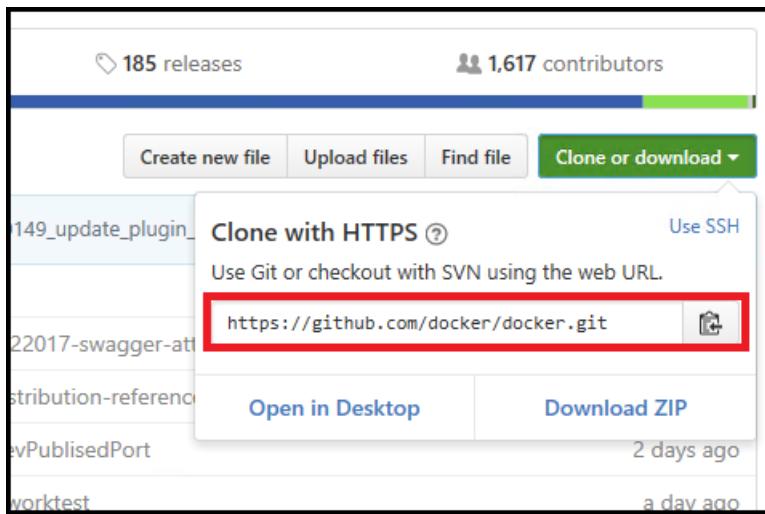
You can open code into Visual Studio in any of the following ways:

- On the Visual Studio menu bar, choose **File** > **Open** > **Folder**, and then browse to the code location.
- On the context (right-click) menu of a folder containing code, choose the **Open in Visual Studio** command.
- Choose the **Open Folder** link on the Visual Studio **Start Page**.
- Choose the **Open Folder** link on the start window.
- If you are a keyboard user, press **Ctrl+Shift+Alt+O** in Visual Studio.
- Open code from a cloned GitHub repo.

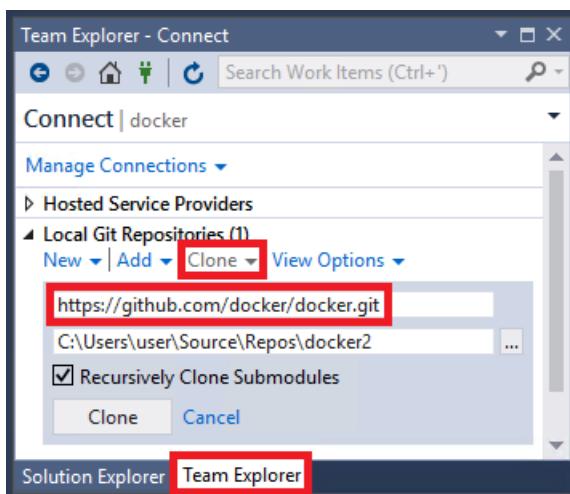
To open code from a cloned GitHub repo

The following example shows how to clone a GitHub repo and then open its code in Visual Studio. To follow this procedure, you must have a GitHub account and Git for Windows installed on your system. See [Signing up for a new GitHub account](#) and [Git for Windows](#) for more information.

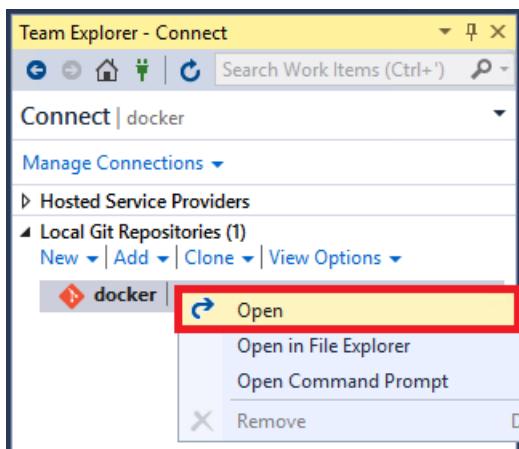
1. Go to the repo you want to clone on GitHub.
2. Choose the **Clone or Download** button and then choose the **Copy to Clipboard** button in the dropdown menu to copy the secure URL for the GitHub repo.



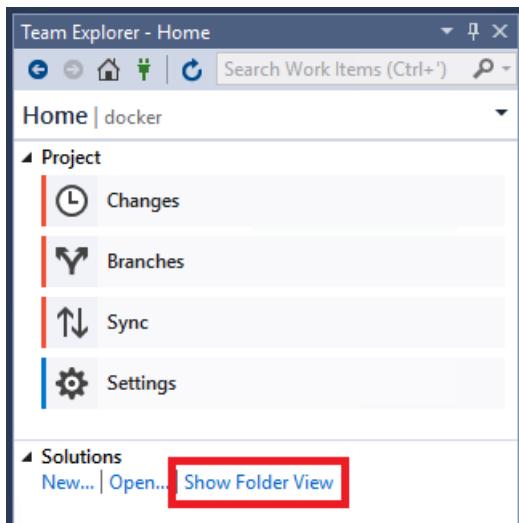
3. In Visual Studio, choose the **Team Explorer** tab to open **Team Explorer**. If you don't see the tab, open it from **View > Team Explorer**.
4. In Team Explorer, under the **Local Git Repositories** section, choose the **Clone** command and then paste the URL of the GitHub page into the text box.



5. Choose the **Clone** button to clone the project's files to a local Git repository. Depending on the size of the repo, this process could take several minutes.
6. After the repo has been cloned to your system, in **Team Explorer**, choose the **Open** command on the context (right-click) menu of the newly cloned repo.



7. Choose the **Show Folder View** command to view the files in **Solution Explorer**.



You can now browse folders and files in the cloned repo, and view and search the code in the Visual Studio code editor, complete with syntax colorization and other features.

Run and debug your code

You can debug your code in Visual Studio without a project or solution! To debug some languages, you may need to specify a valid *startup file* in the codebase, such as a script, executable, or project. The drop-down list box next to the **Start** button on the toolbar lists all of the startup items that Visual Studio detects, as well as items you specifically designate. Visual Studio runs this code first when you debug your code.

Configuring your code to run in Visual Studio differs depending on what kind of code it is, and what the build tools are.

Codebases that use MSBuild

MSBuild-based codebases can have multiple build configurations that appear in the **Start** button's drop-down list. Select the file that you want to use as the startup item, and then choose the **Start** button to begin debugging.

NOTE

For C# and Visual Basic codebases, you must have the **.NET desktop development** workload installed. For C++ codebases, you must have the **Desktop development with C++** workload installed.

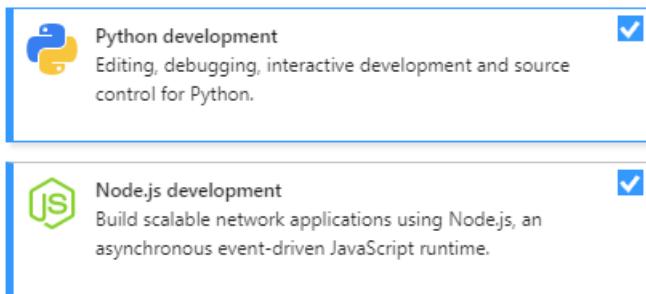
Codebases that use custom build tools

If your codebase uses custom build tools, then you must tell Visual Studio how to build your code using *build tasks* that are defined in a *.json* file. For more information, see [Customize build and debug tasks](#).

Codebases that contain Python or JavaScript code

If your codebase contains Python or JavaScript code, you don't have to configure any *.json* files, but you do have to install the corresponding workload. You must also configure the startup script:

1. Install the [Node.js development](#) or [Python development](#) workload by choosing **Tools > Get Tools and Features**, or by closing Visual Studio and running the Visual Studio Installer.



2. In **Solution Explorer**, on the right-click or context menu of a JavaScript or Python file, choose the **Set as Startup Item** command.

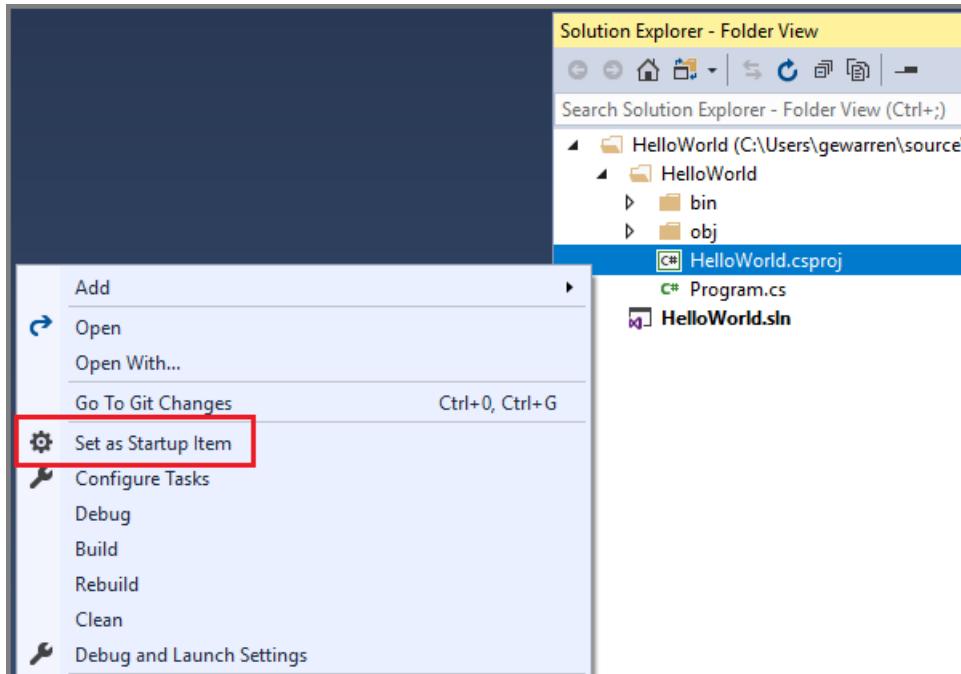
3. Choose the **Start** button to begin debugging.

Codebases that contain C++ code

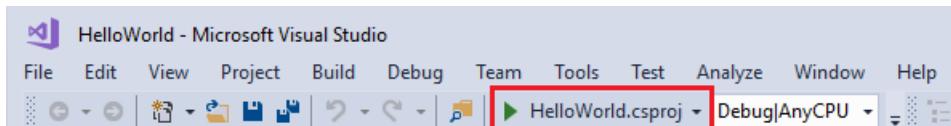
For information about opening C++ code without solutions or projects in Visual Studio, see [Open Folder projects for C++](#).

Codebases that contain a Visual Studio project

If your code folder contains a Visual Studio project, you can designate the project as the startup item.



The **Start** button's text changes to reflect that the project is the startup item.



See also

- [Customize build and debug tasks](#)
- [Open Folder projects for C++](#)
- [CMake projects in C++](#)
- [Writing code in the code and text editor](#)

Customize build and debug tasks for "Open Folder" development

10/18/2019 • 9 minutes to read • [Edit Online](#)

Visual Studio knows how to run many different languages and codebases, but it doesn't know how to run everything. If you [opened a code folder](#) in Visual Studio, and Visual Studio knows how to run your code, you can run it right away without any additional configuration.

If the codebase uses custom build tools that Visual Studio doesn't recognize, you need to provide some configuration details to run and debug the code in Visual Studio. You instruct Visual Studio how to build your code by defining *build tasks*. You can create one or more build tasks to specify all the items a language needs to build and run its code. You can also create arbitrary tasks that can do nearly anything you want. For example, you can create a task to list the contents of a folder or to rename a file.

Customize your project-less codebase by using the following *json* files:

FILE NAME	PURPOSE
<i>tasks.vs.json</i>	Specify custom build commands and compiler switches, and arbitrary (non-build related) tasks. Accessed via the Solution Explorer right-click menu item Configure Tasks .
<i>launch.vs.json</i>	Specify command-line arguments for debugging. Accessed via the Solution Explorer right-click menu item Debug and Launch Settings .

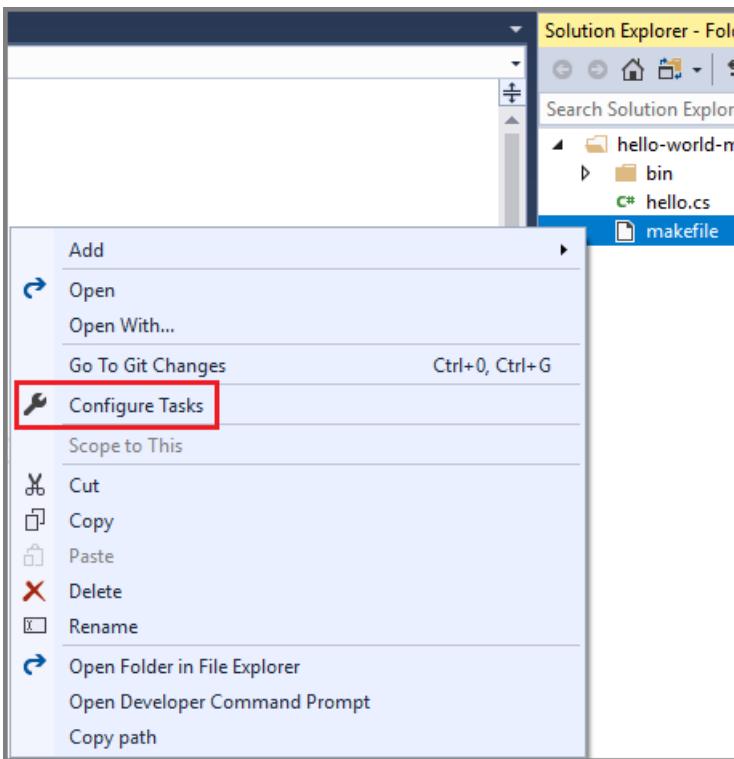
These *json* files are located in a hidden folder called *.vs* in the root folder of your codebase. The *tasks.vs.json* and *launch.vs.json* files are created by Visual Studio on an as-needed basis when you choose either **Configure Tasks** or **Debug and Launch Settings** on a file or folder in **Solution Explorer**. These *json* files are hidden because users generally don't want to check them into source control. However, if you want to be able to check them into source control, drag the files into the root of your codebase, where they are visible.

TIP

To view hidden files in Visual Studio, choose the **Show All Files** button on the **Solution Explorer** toolbar.

Define tasks with *tasks.vs.json*

You can automate build scripts or any other external operations on the files you have in your current workspace by running them as tasks directly in the IDE. You can configure a new task by right-clicking on a file or folder and selecting **Configure Tasks**.



This creates (or opens) the `tasks.vs.json` file in the `.vs` folder. You can define a build task or arbitrary task in this file, and then invoke it using the name you gave it from the **Solution Explorer** right-click menu.

Custom tasks can be added to individual files, or to all files of a specific type. For instance, NuGet package files can be configured to have a "Restore Packages" task, or all source files can be configured to have a static analysis task, such as a linter for all `.js` files.

Define custom build tasks

If your codebase uses custom build tools that Visual Studio doesn't recognize, then you cannot run and debug the code in Visual Studio until you complete some configuration steps. Visual Studio provides *build tasks* where you can tell Visual Studio how to build, rebuild, and clean your code. The `tasks.vs.json` build task file couples the Visual Studio inner development loop to the custom build tools used by your codebase.

Consider a codebase that consists of a single C# file called `hello.cs`. The `makefile` for such a codebase might look like this:

```
build: directory hello.exe

hello.exe: hello.cs
    csc -debug hello.cs /out:bin\hello.exe

clean:
    del bin\hello.exe bin\hello.pdb

rebuild: clean build

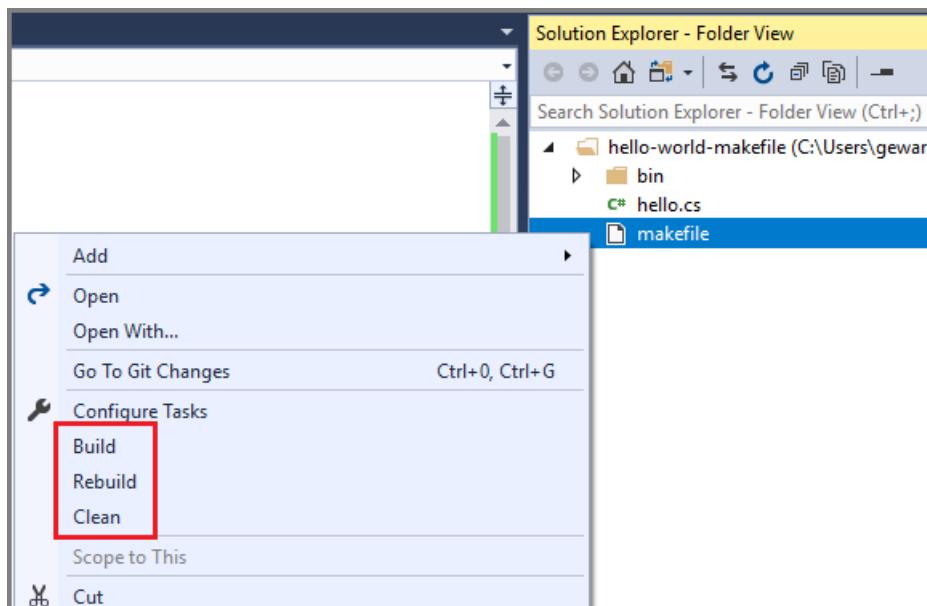
directory: bin

bin:
    md bin
```

For such a `makefile` that contains build, clean, and rebuild targets, you can define the following `tasks.vs.json` file. It contains three build tasks for building, rebuilding, and cleaning the codebase, using NMAKE as the build tool.

```
{
  "version": "0.2.1",
  "outDir": "${workspaceRoot}\\bin\\",
  "tasks": [
    {
      "taskName": "makefile-build",
      "appliesTo": "makefile",
      "type": "launch",
      "contextType": "build",
      "command": "nmake",
      "args": [ "build" ],
      "envVars": {
        "VSCMD_START_DIR": "${workspaceRoot}\\"
      }
    },
    {
      "taskName": "makefile-clean",
      "appliesTo": "makefile",
      "type": "launch",
      "contextType": "clean",
      "command": "nmake",
      "args": [ "clean" ],
      "envVars": {
        "VSCMD_START_DIR": "${workspaceRoot}\\"
      }
    },
    {
      "taskName": "makefile-rebuild",
      "appliesTo": "makefile",
      "type": "launch",
      "contextType": "rebuild",
      "command": "nmake",
      "args": [ "rebuild" ],
      "envVars": {
        "VSCMD_START_DIR": "${workspaceRoot}\\"
      }
    }
  ]
}
```

After you define build tasks in `tasks.json`, additional right-click menu (context menu) items are added to the corresponding files in **Solution Explorer**. For this example, "build", "rebuild", and "clean" options are added to the context menu of any *makefile* files.



NOTE

The commands appear in the context menu under the **Configure Tasks** command due to their `contextType` settings. "build", "rebuild", and "clean" are build commands, so they appear in the build section in the middle of the context menu.

When you select one of these options, the task executes. Output appears in the **Output** window, and build errors appear in the **Error List**.

Define arbitrary tasks

You can define arbitrary tasks in the `tasks.json` file, to do just about anything you want. For example, you can define a task to display the name of the currently selected file in the **Output** window, or to list the files in a specified directory.

The following example shows a `tasks.json` file that defines a single task. When invoked, the task displays the filename of the currently selected `.js` file.

```
{  
  "version": "0.2.1",  
  "tasks": [  
    {  
      "taskName": "Echo filename",  
      "appliesTo": "*.js",  
      "type": "default",  
      "command": "${env.COMSPEC}",  
      "args": [ "echo ${file}" ]  
    }  
  ]  
}
```

- `taskName` specifies the name that appears in the right-click menu.
- `appliesTo` specifies which files the command can be performed on.
- The `command` property specifies the command to invoke. In this example, the `COMSPEC` environment variable is used to identify the command line interpreter, typically `cmd.exe`.
- The `args` property specifies the arguments to be passed to the invoked command.
- The `${file}` macro retrieves the selected file in **Solution Explorer**.

After saving `tasks.json`, you can right-click on any `.js` file in the folder and choose **Echo filename**. The file name is displayed in the **Output** window.

NOTE

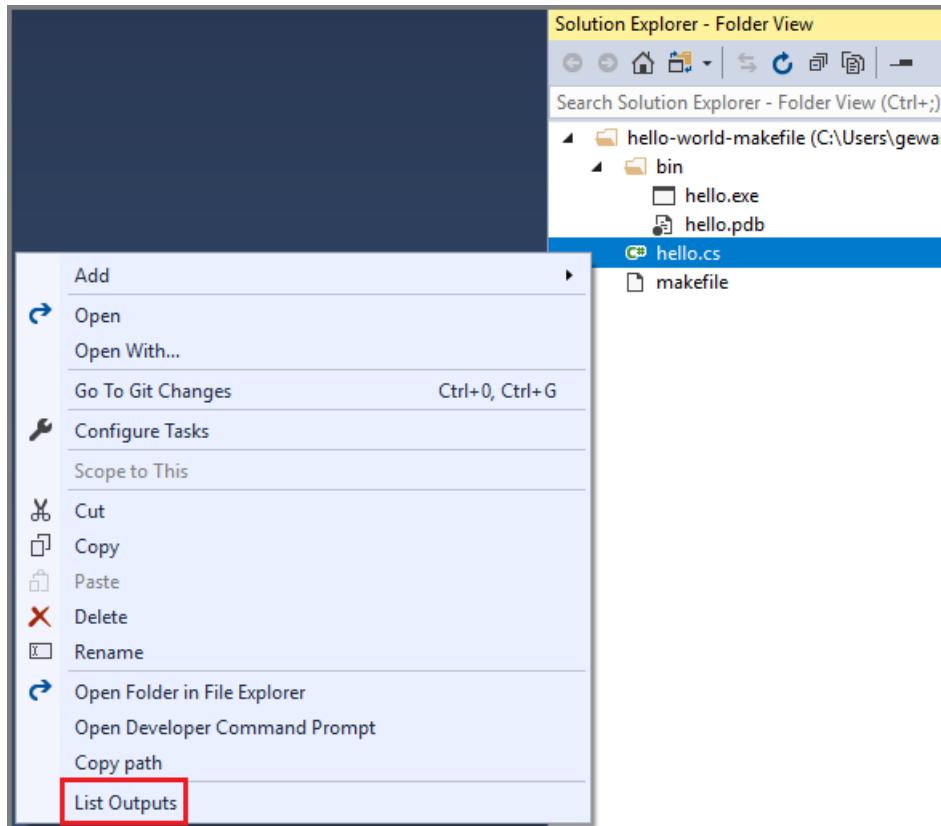
If your codebase doesn't contain a `tasks.json` file, you can create one by choosing **Configure Tasks** from the right-click or context menu of a file in **Solution Explorer**.

The next example defines a task that lists the files and subfolders of the `bin` directory.

```
{
  "version": "0.2.1",
  "outDir": "${workspaceRoot}\\bin\\",
  "tasks": [
    {
      "taskName": "List Outputs",
      "appliesTo": "*",
      "type": "default",
      "command": "${env.COMSPEC}",
      "args": [ "dir ${outDir}" ]
    }
  ]
}
```

- `${outDir}` is a custom macro that is first defined before the `tasks` block. It is then called in the `args` property.

This task applies to all files. When you open the context menu on any file in **Solution Explorer**, the task's name **List Outputs** appears at the bottom of the menu. When you choose **List Outputs**, the contents of the *bin* directory are listed in the **Output** window in Visual Studio.



Settings scope

Multiple `tasks.json` files can exist at the root and subdirectories of a codebase. This design enables the flexibility to have different behavior in different subdirectories of the codebase. Visual Studio aggregates or overrides settings throughout the codebase, prioritizing files in the following order:

- Settings files in the root folder's `.vs` directory.
- The directory where a setting is being computed.
- The current directory's parent directory, all the way up to the root directory.
- Settings files in the root directory.

These aggregation rules apply to `tasks.json`. For information on how settings in other files are aggregated, see the corresponding section for that file in this article.

Properties for `tasks.json`

This section describes some of the properties you can specify in `tasks.json`.

appliesTo

You can create tasks for any file or folder by specifying its name in the `appliesTo` field, for example

```
"appliesTo": "hello.js"
```

The following file masks can be used as values:

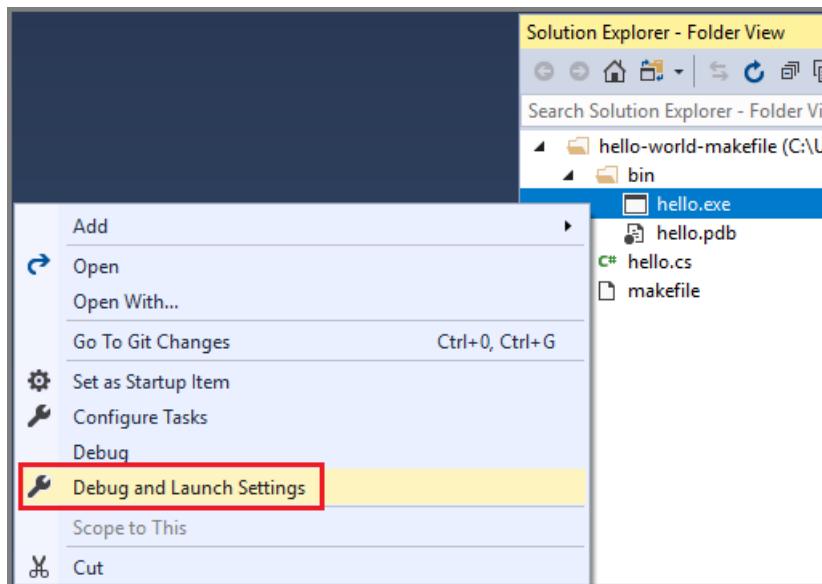
<code>"*"</code>	task is available to all files and folders in the workspace
<code>"*/"</code>	task is available to all folders in the workspace
<code>"*.js"</code>	task is available to all files with the extension <code>.js</code> in the workspace
<code>"/*.js"</code>	task is available to all files with the extension <code>.js</code> in the root of the workspace
<code>"src/*/"</code>	task is available to all subfolders of the <code>src</code> folder
<code>"makefile"</code>	task is available to all <code>makefile</code> files in the workspace
<code>"/makefile"</code>	task is available only to the <code>makefile</code> in the root of the workspace

Macros for `tasks.json`

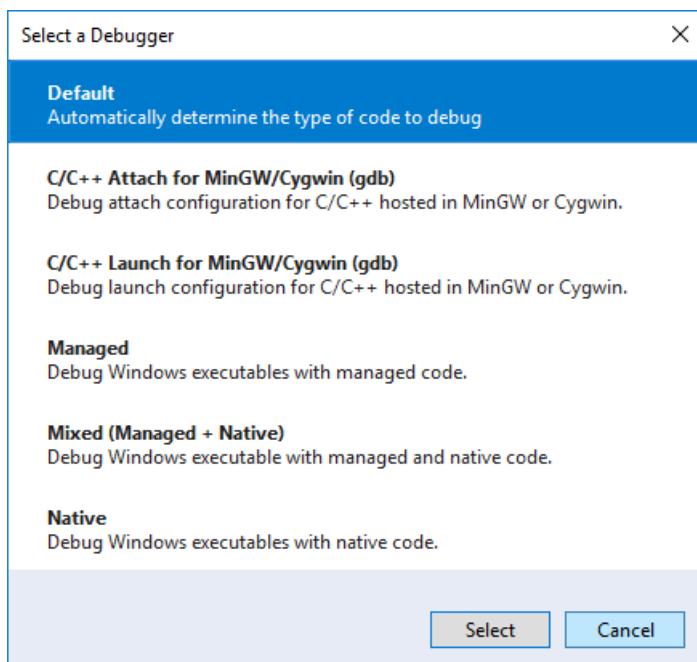
<code> \${env.<VARIABLE>}</code>	Specifies any environment variable (for example, <code> \${env.PATH}</code> , <code> \${env.COMSPEC}</code> and so on) that is set for the developer command prompt. For more information, see Developer command prompt for Visual Studio .
<code> \${workspaceRoot}</code>	The full path to the workspace folder (for example, <code>C:\sources\hello</code>)
<code> \${file}</code>	The full path of the file or folder selected to run this task against (for example, <code>C:\sources\hello\src\hello.js</code>)
<code> \${relativeFile}</code>	The relative path to the file or folder (for example, <code>src\hello.js</code>)
<code> \${fileBasename}</code>	The name of the file without path or extension (for example, <code>hello</code>)
<code> \${fileDirname}</code>	The full path to the file, excluding the filename (for example, <code>C:\sources\hello\src</code>)
<code> \${fileExtname}</code>	The extension of the selected file (for example, <code>.js</code>)

Configure debugging with `launch.json`

- To configure your codebase for debugging, in **Solution Explorer** choose the **Debug and Launch Settings** menu item from the right-click or context menu of your executable file.



2. In the **Select a Debugger** dialog box, choose an option, and then choose the **Select** button.

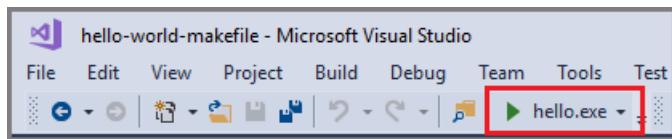


If the *launch.json* file doesn't already exist, it is created.

```
{  
    "version": "0.2.1",  
    "defaults": {},  
    "configurations": [  
        {  
            "type": "default",  
            "project": "bin\\hello.exe",  
            "name": "hello.exe"  
        }  
    ]  
}
```

3. Next, right-click on the executable file in **Solution Explorer**, and choose **Set as Startup Item**.

The executable is designated as the startup item for your codebase, and the debugging **Start** button's title changes to reflect the name of your executable.



When you choose **F5**, the debugger launches and stops at any breakpoint you may have already set. All the familiar debugger windows are available and functional.

IMPORTANT

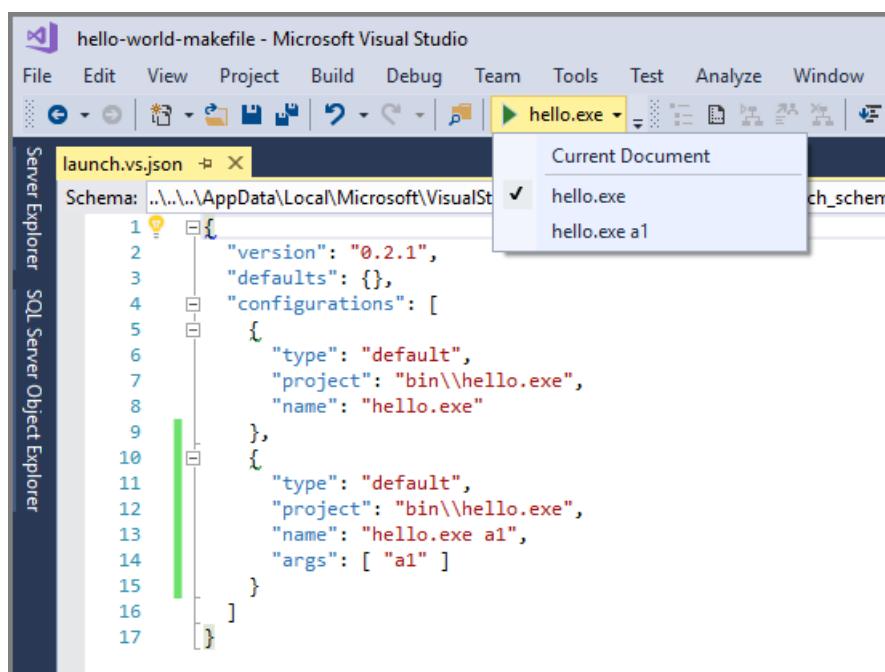
For additional details about custom build and debug tasks in C++ open folder projects, see [Open Folder support for C++ build systems in Visual Studio](#).

Specify arguments for debugging

You can specify command-line arguments to pass in for debugging in the *launch.json* file. Add the arguments in the `args` array, as shown in the following example:

```
{
  "version": "0.2.1",
  "defaults": {},
  "configurations": [
    {
      "type": "default",
      "project": "bin\\hello.exe",
      "name": "hello.exe"
    },
    {
      "type": "default",
      "project": "bin\\hello.exe",
      "name": "hello.exe a1",
      "args": [ "a1" ]
    }
  ]
}
```

When you save this file, the name of the new configuration appears in the debug target drop-down list, and you can select it to start the debugger. You can create as many debug configurations as you like.



NOTE

The `configurations` array property in `launch.vs.json` is read from two file locations—the root directory for the codebase, and the `.vs` directory. If there is a conflict, priority is given to the value in `.vs\launch.vs.json`.

Additional settings files

In addition to the three `json` files described in this topic, Visual Studio also reads settings from some additional files, if they exist in your codebase.

.vscode\settings.json

Visual Studio reads limited settings from a file named `settings.json`, if it is in a directory named `.vscode`. This functionality is provided for codebases that have previously been developed in Visual Studio Code. Currently, the only setting that is read from `.vscode\settings.json` is `files.exclude`, which filters files visually in Solution Explorer and from some search tools.

You can have any number of `.vscode\settings.json` files in your codebase. Settings read from this file are applied to the parent directory of `.vscode` and all of its subdirectories.

.gitignore

`.gitignore` files are used to tell Git which files to ignore; that is, which files and directories you don't want to check in. `.gitignore` files are usually included as part of a codebase so that the settings can be shared with all developers of the codebase. Visual Studio reads patterns in `.gitignore` files to filter items visually and from some search tools.

Settings read from the `.gitignore` file are applied to its parent directory and all subdirectories.

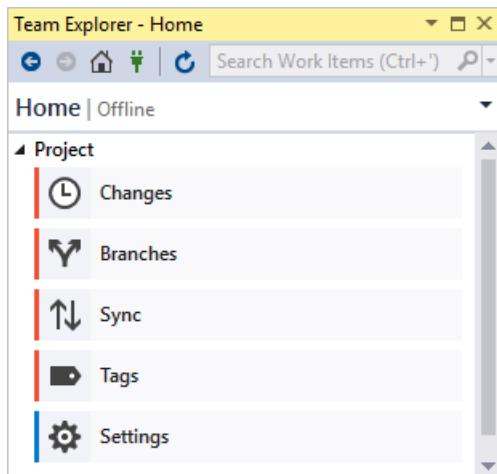
See also

- [Develop code without projects or solutions](#)
- [Open Folder projects for C++](#)
- [CMake projects for C++](#)
- [NMAKE reference](#)
- [Features of the code editor](#)

Connect to projects in Team Explorer

11/26/2019 • 2 minutes to read • [Edit Online](#)

Use the **Team Explorer** tool window to coordinate your code efforts with other team members to develop a project, and to manage work that's assigned to you, your team, or your projects. **Team Explorer** connects Visual Studio to Git and GitHub repositories, Team Foundation version control (TFVC) repositories, and projects hosted on [Azure DevOps Services](#) or an on-premises [Azure DevOps Server](#) (formerly known as TFS). You can manage source code, work items, and builds.

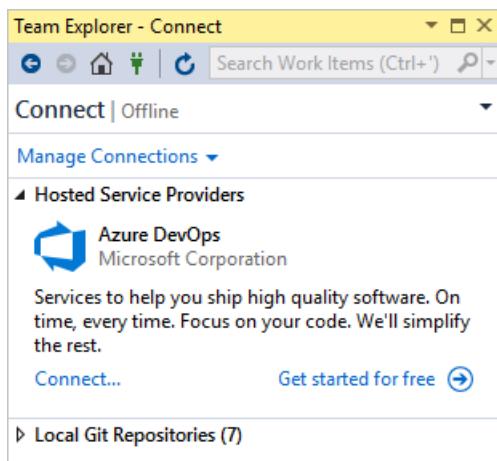


TIP

If you open Visual Studio and **Team Explorer** doesn't appear, open it by choosing **View > Team Explorer** from the menu bar.

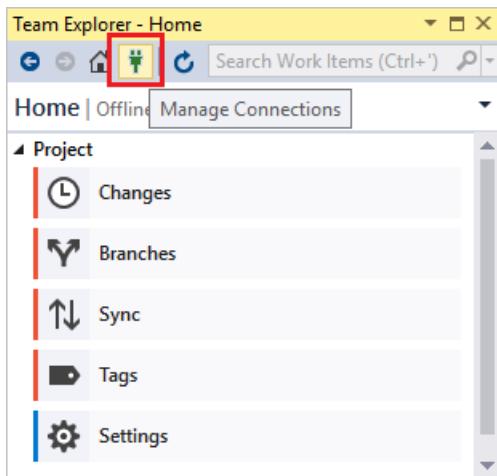
Connect to a project or repository

Connect to a project or repository on the **Connect** page.

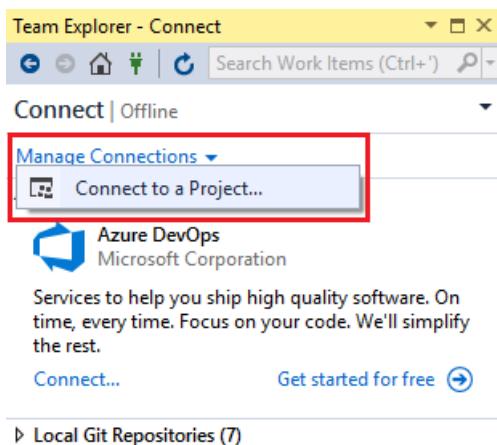


To connect to a project:

1. Open the **Connect** page by choosing the **Manage Connections** icon.



2. On the **Connect** page, choose **Manage Connections > Connect to a project**.



TIP

If you need to create a new project or add users to a project, see [Create a project \(Azure DevOps\)](#) and [Add users to a project or team \(Azure DevOps\)](#).

See also

- [Team Explorer reference](#)
- [Connect to a project \(Azure DevOps\)](#)

Features of the code editor

10/18/2019 • 9 minutes to read • [Edit Online](#)

The Visual Studio editor provides many features that make it easier for you to write and manage your code and text. You can expand and collapse different blocks of code by using outlining. You can learn more about the code by using IntelliSense, the **Object Browser**, and the Call Hierarchy. You can find code by using features such as **Go To**, **Go To Definition**, and **Find All References**. You can insert blocks of code with code snippets, and you can generate code by using features such as **Generate From Usage**. If you have never used the Visual Studio editor before, see [Learn to use the code editor](#).

NOTE

This topic applies to Visual Studio on Windows. For Visual Studio for Mac, see [Source editor \(Visual Studio for Mac\)](#).

You can view your code in a number of different ways. By default, **Solution Explorer** shows your code organized by files. You can click on the **Class View** tab at the bottom of the window to view your code organized by classes.

You can search and replace text in single or multiple files. For more information, see [Find and replace text](#). You can use regular expressions to find and replace text. For more information, see [Use regular expressions in Visual Studio](#).

The different Visual Studio languages offer different sets of features, and in some cases the features behave differently in different languages. Many of these differences are specified in the descriptions of the features, but for more information you can see the sections on specific Visual Studio languages.

Editor features

Syntax Coloring

Some syntax elements of code and markup files are colored differently to distinguish them. For example, keywords (such as `using` in C# and `Imports` in Visual Basic) are one color, but types (such as `Console` and `Uri`) are another color. Other syntax elements are also colorized, such as string literals and comments. C++ uses color to differentiate among types, enumerations, and macros, among other tokens.

You can see the default color for each type, and you can change the color for any specific syntax element in the [Fonts and Colors, Environment, Options dialog box](#), which you can open from the **Tools** menu.

Error and Warning Marks	<p>As you add code and build your solution, you may see (a) different-colored wavy underlines (known as squiggles) or (b) light bulbs appearing in your code. Red squiggles denote syntax errors, blue denotes compiler errors, green denotes warnings, and purple denotes other types of errors. Quick Actions suggest fixes for problems and make it easy to apply the fix.</p> <p>You can see the default color for each error and warning squiggle in the Tools > Options > Environment > Fonts and Colors dialog box. Look for Syntax Error, Compiler Error, Warning, and Other Error.</p>
Brace Matching	<p>When the insertion point is placed on an open brace in a code file, both it and the closing brace are highlighted. This feature gives you immediate feedback on misplaced or missing braces. You can turn brace matching on or off with the Automatic Delimiter Highlighting setting (Tools > Options > Text Editor). You can change the highlight color in the Fonts and Colors setting (Tools > Options > Environment). Look for Brace Matching (Highlight) or Brace Matching (Rectangle).</p>
Structure Visualizer	<p>Dotted lines connect matching braces in code files, making it easier to see opening and closing brace pairs. This can help you find code in your codebase more quickly. You can turn these lines on or off with the Show structure guidelines in the Display section of the Tools > Options > Text Editor > General page.</p>
Line Numbers	<p>Line numbers can be displayed in the left margin of the code window. They are not displayed by default. You can turn this option on in the Text Editor All Languages settings (Tools > Options > Text Editor > All Languages). You can display line numbers for individual programming languages by changing the settings for those languages (Tools > Options > Text Editor > <language>). For line numbers to print, you must select Include line numbers in the Print dialog box.</p>
Change Tracking	<p>The color of the left margin allows you to keep track of the changes you have made in a file. Changes you have made since the file was opened but not saved are denoted by a yellow bar on the left margin (known as the selection margin). After you have saved the changes (but before closing the file), the bar turns green. If you undo a change after you have saved the file, the bar turns orange. To turn this feature off and on, change the Track changes option in the Text Editor settings (Tools > Options > Text Editor).</p>
Selecting Code and Text	<p>You can select text either in the standard continuous stream mode or in box mode, in which you select a rectangular portion of text instead of a set of lines. To make a selection in box mode, press Alt as you drag the mouse over the selection (or press Alt+Shift+<arrow key>). The selection includes all of the characters within the rectangle defined by the first character and the last character in the selection. Anything typed or pasted into the selected area is inserted at the same point on each line.</p>

Zoom	You can zoom in or out in any code window by pressing and holding the Ctrl key and moving the scroll wheel on the mouse (or Ctrl+Shift+ to increase and Ctrl+Shift- to decrease). You can also use the Zoom box in the lower left corner of the code window to set a specific zoom percentage. The zoom feature does not work in tool windows.
Virtual Space	By default, lines in Visual Studio editors end after the last character, so that the Right Arrow key at the end of a line moves the cursor to the beginning of the next line. In some other editors a line does not end after the last character, and you can place your cursor anywhere on the line. You can enable virtual space in the editor in the Tools > Options > Text Editor > All Languages settings. Note that you can enable either Virtual Space or Word Wrap , but not both.
Printing	<p>You can use the options in the Print dialog box to include line numbers or hide collapsed regions of code when you print a file. In the Page Setup dialog box, you can also choose to print the full path and the name of the file by choosing Page header.</p> <p>You can set color printing options in the Tools > Options > Environment > Fonts and Colors dialog box. Choose Printer in the Show settings for list to customize color printing. You can specify different colors for printing a file than for editing a file.</p>
Global Undo and Redo	The Undo Last Global Action and Redo Last Global Action commands on the Edit menu undo or redo global actions that affect multiple files. Global actions include renaming a class or namespace, performing a find-and-replace operation across a solution, refactoring a database, or any other action that changes multiple files. You can apply the global undo and redo commands to actions in the current Visual Studio session, even after you close the solution in which an action was applied.

Advanced editing features

You can find a number of advanced features on the **Edit > Advanced** menu on the toolbar. Not all of these features are available for all types of code files.

Format Document	Sets the proper indentation of lines of code and moves curly braces to separate lines in the document.
Format Selection	Sets the proper indentation of lines of code and moves curly braces to separate lines in the selection.
Tabify Selected Lines	Changes leading spaces to tabs where appropriate.

Untabify Selected Lines	Changes leading tabs to spaces. If you want to convert all the spaces in your file to tabs (or all the tabs to spaces), you can use the <code>Edit.ConvertSpacesToTabs</code> and <code>Edit.ConvertTabsToSpaces</code> commands. These commands do not appear in Visual Studio menus, but you can call them from the Quick Access window or the command window.
Make Uppercase	Changes all characters in the selection to uppercase, or if there is no selection, changes the character at the insertion point to uppercase. Shortcut: Ctrl+Shift+U .
Make Lowercase	Changes all characters in the selection to lowercase, or if there is no selection, changes the character at the insertion point to lowercase. Shortcut: Ctrl+U .
Move selected Lines Up	Moves the selected line up one line. Shortcut: Alt+Up Arrow .
Move Selected Lines Down	Moves the selected line down one line. Shortcut: Alt+Down Arrow .
Delete Horizontal White Space	Deletes tabs or spaces at the end of the current line. Shortcut: Ctrl+K, Ctrl+\
View White Space	Displays spaces as raised dots, and tabs as arrows. The end of a file is displayed as a rectangular glyph. If Tools > Options > Text Editor > All Languages > Word Wrap > Show visible glyphs for word wrap is selected, that glyph is also displayed.
Word Wrap	Causes all the lines in a document to be visible in the code window. You can turn word wrap off and on in the Text Editor All Languages settings (Tools > Options > Text Editor > All Languages).
Comment Selection	Adds comment characters to the selection or the current line. Shortcut: Ctrl+K, Ctrl+C
Uncomment Selection	Removes comment characters from the selection or the current line. Shortcut: Ctrl+K, Ctrl+U
Increase Line Indent	Adds a tab (or the equivalent spaces) to the selected lines or the current line.
Decrease Line Indent	Removes a tab (or the equivalent spaces) from the selected lines or the current line.
Select Tag	In a document that contains tags (for example, XML or HTML), selects the tag.
Select Tag Content	In a document that contains tags (for example, XML or HTML), selects the content.

Navigate and find code

You can move around in the code editor in several different ways, including navigating backwards and forwards to previous insertion points, viewing the definition of a type or member, and jumping to a specific method using the navigation bar. For more information see [Navigate code](#).

Find references in your code base

To find where particular code elements are referenced throughout your codebase, you can use the **Find All References** command or press **Shift+F12**. Also, when you click on a type or member, the **reference highlighting** feature automatically highlights all references to that type or member. For more information, see [Find references in your code](#).

Customize the editor

You can share your Visual Studio settings with another developer, have your settings conform to a standard, or return to Visual Studio default settings by using the **Import and Export Settings Wizard** command on the **Tools** menu. In the **Import and Export Settings Wizard**, you can change selected general settings or language and project-specific settings.

To define new hotkeys or redefine existing hotkeys, go to **Tools > Options > Environment > Keyboard**. For more information about hotkeys, see [Default keyboard shortcuts](#).

For JavaScript-specific editor options, see [JavaScript editor options](#).

See also

- [Source editor \(Visual Studio for Mac\)](#)
- [Visual Studio IDE](#)
- [Get started with C++ in Visual Studio](#)
- [Get started with C# and ASP.NET in Visual Studio](#)
- [Get started with Python in Visual Studio](#)

Find and replace text

10/18/2019 • 5 minutes to read • [Edit Online](#)

You can find and replace text in the Visual Studio editor by using [Find and Replace](#) (**Ctrl+F** or **Ctrl+H**) or [Find/Replace in Files](#) (**Ctrl+Shift+F** or **Ctrl+Shift+H**). You can also find and replace only *some* instances of a pattern by using *multi-caret selection*.

TIP

If you're renaming code symbols such as variables and methods, it's better to [refactor](#) them than to use find-and-replace. Refactoring is intelligent and understands scope, whereas find-and-replace blindly replaces all instances.

Find-and-replace functionality is available in the editor, in certain other text-based windows such as the **Find Results** windows, in designer windows such as the XAML designer and Windows Forms designer, and in tool windows.

You can scope searches to the current document, the current solution, or a custom set of folders. You can also specify a set of file name extensions for multi-file searches. Customize search syntax by using .NET [regular expressions](#).

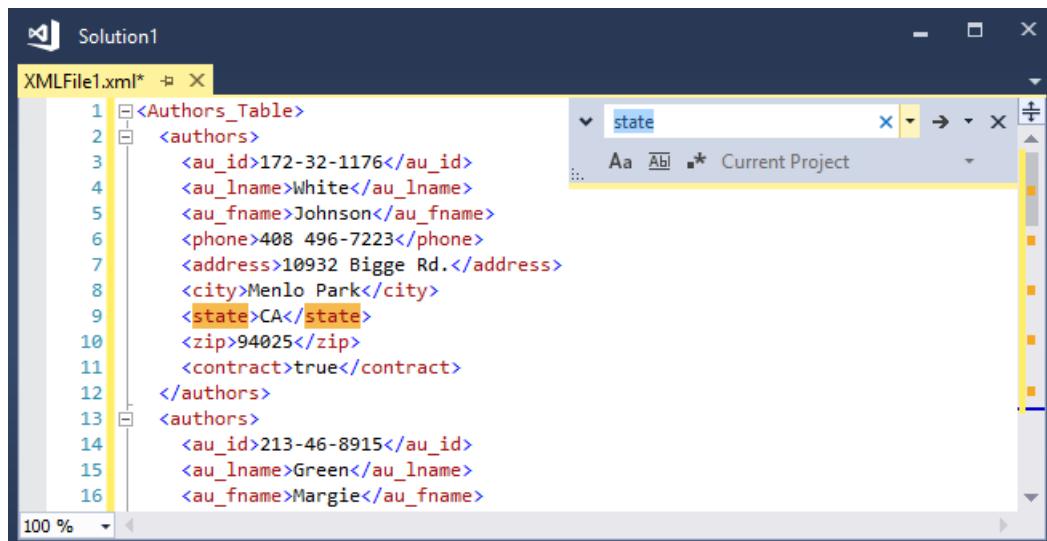
TIP

The [Find/Command](#) box is available as a toolbar control, but isn't visible by default. To display the **Find/Command** box, select **Add or Remove Buttons** on the **Standard** toolbar, and then select **Find**.

Find and Replace control

- Press **Ctrl+F** as a shortcut to *find* a string in the current file.
- Press **Ctrl+H** as a shortcut to *find and replace* a string in the current file.

The **Find and Replace** control appears in the upper right corner of the code editor window. It immediately highlights every occurrence of the given search string in the current document. You can navigate from one occurrence to another by choosing the **Find Next** button or the **Find Previous** button on the search control.



You can access replacement options by choosing the button next to the **Find** text box. To make one replacement

at a time, choose the **Replace Next** button next to the **Replace** text box. To replace all matches, choose the **Replace All** button.

To change the highlight color for matches, choose the **Tools** menu, select **Options**, and then choose **Environment**, and select **Fonts and Colors**. In the **Show settings for** list, select **Text Editor**, and then in the **Display items** list, select **Find Highlight (Extension)**.

Search tool windows

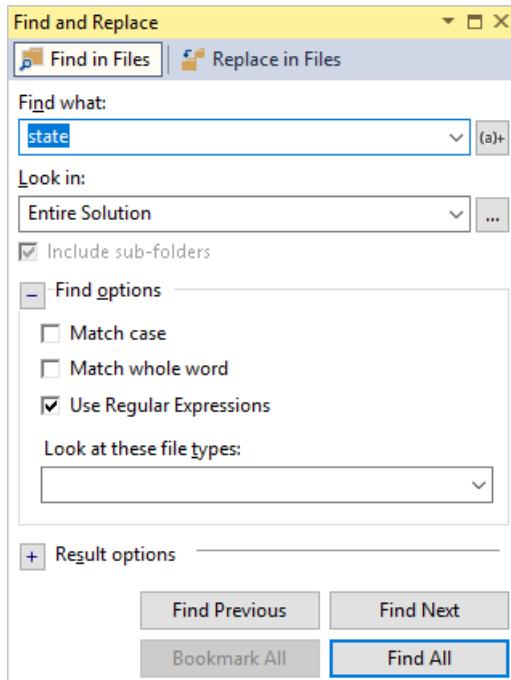
You can use the **Find** control in code or text windows, such as **Output** windows and **Find Results** windows, by selecting **Edit > Find and Replace** or pressing **Ctrl+F**.

A version of the **Find** control is also available in some tool windows. For example, you can filter the list of controls in the **Toolbox** window by entering text in the search box. Other tool windows that allow you to search their contents include **Solution Explorer**, the **Properties** window, and **Team Explorer**.

Find in Files and Replace in Files

- Press **Ctrl+Shift+F** as a shortcut to *find* a string in multiple files.
- Press **Ctrl+Shift+H** as a shortcut to *find and replace* a string in multiple files.

Find/Replace in Files works like the **Find and Replace** control, except that you can define a scope for your search. Not only can you search the current open file in the editor, but also all open documents, the entire solution, the current project, and selected folder sets. You can also search by file name extension. To access the **Find/Replace in Files** dialog box, select **Find and Replace** on the **Edit** menu (or press **Ctrl+Shift+F**).



Find Results

When you choose **Find All**, a **Find Results** window opens and lists the matches for your search. Selecting a result in the list displays the associated file and highlights the match. If the file is not already open for editing, it is opened in a preview tab in the right side of the tab well. You can use the **Find** control to search through the **Find Results** list.

Create custom search folder sets

You can define a search scope by choosing the **Choose Search Folders** button (it looks like ...) next to the **Look in** box. In the **Choose Search Folders** dialog box, you can specify a set of folders to search, and you can save the specification so that you can reuse it later.

TIP

If you've mapped a remote machine's drive to your local machine, you can specify folders to search on the remote machine.

Create custom component sets

You can define component sets as your search scope by choosing the **Edit Custom Component Set** button next to the **Look in** box. You can specify installed .NET or COM components, Visual Studio projects that are included in your solution, or any assembly or type library (.dll, .tlb, .olb, .exe, or .ocx). To search references, select the **Look in references** box.

Multi-caret selection

NOTE

This section applies to Visual Studio on Windows. For Visual Studio for Mac, see [Block selection](#).

Introduced in Visual Studio 2017 version 15.8

Use *multi-caret selection* to make the same edit in two or more places at the same time. For example, you can insert the same text or modify existing text in multiple locations at the same time.

In the following screenshot, `-0000` is selected in three locations; if the user presses **Delete**, all three selections are deleted:



To select multiple caret, click or make first text selection as usual, and then press **Alt** while you click or select text in each additional location. You can also automatically add matching text as additional selections, or select a box

of text to edit identically on each line.

TIP

If you've selected **Alt** as the modifier key for mouse-click Go to Definition in **Tools > Options**, multi-caret select is disabled.

Commands

Use the following keys and actions for multi-caret selection behaviors:

SHORTCUT	ACTION
Ctrl+Alt + click	Add a secondary caret
Ctrl+Alt + double-click	Add a secondary word selection
Ctrl+Alt + click + drag	Add a secondary selection
Shift+Alt+ .	Add the next matching text as a selection
Ctrl+Shift+Alt+ ,	Add all matching text as selections
Shift+Alt+ ,	Remove last selected occurrence
Ctrl+Shift+Alt+ .	Skip next matching occurrence
Alt + click	Add a box selection
Esc or click	Clear all selections

Some of the commands are also available on the **Edit** menu, under **Multiple Carets**:



See also

- [Use regular expressions in Visual Studio](#)
- [Refactor code in Visual Studio](#)
- [Block selection \(Visual Studio for Mac\)](#)

Use regular expressions in Visual Studio

10/18/2019 • 6 minutes to read • [Edit Online](#)

Visual Studio uses [.NET regular expressions](#) to find and replace text.

Regular expression examples

The following table contains some regular expression characters, operators, constructs, and pattern examples.

For a more complete reference, see [Regular expression language](#).

PURPOSE	EXPRESSION	EXAMPLE
Match any single character (except a line break). For more information, see Any character .	.	<code>a.o</code> matches "aro" in "around" and "abo" in "about" but not "acro" in "across"
Match zero or more occurrences of the preceding expression (match as many characters as possible). For more information, see Match zero or more times .	*	<code>a*r</code> matches "r" in "rack", "ar" in "ark", and "aar" in "aardvark"
Match any character zero or more times.	.*	<code>c.*e</code> matches "cke" in "racket", "comme" in "comment", and "code" in "code"
Match one or more occurrences of the preceding expression (match as many characters as possible). For more information, see Match one or more times .	+	<code>e+d</code> matches "eed" in "feeder" and "ed" in "faded"
Match any character one or more times.	.+	<code>e.+e</code> matches "eede" in "feeder" but finds no matches in "feed"
Match zero or more occurrences of the preceding expression (match as few characters as possible). For more information, see Match zero or more times (lazy match) .	*?	<code>\w*?d</code> matches "fad" and "ed" in "faded" but not the entire word "faded" due to the lazy match
Match one or more occurrences of the preceding expression (match as few characters as possible). For more information, see Match one or more times (lazy match) .	+?	<code>e\w+?d</code> matches "ee" in "asleep" and "ed" in "faded" but finds no matches in "fade"
Anchor the match string to the beginning of a line or string	^	<code>^car</code> matches the word "car" only when it appears at the beginning of a line
Anchor the match string to the end of a line	\r?\$	<code>car\r?\$</code> matches "car" only when it appears at the end of a line

PURPOSE	EXPRESSION	EXAMPLE
Anchor the match string to the end of the file	\$	car\$ matches "car" only when it appears at the end of the file
Match any single character in a set	[abc]	b[abc] matches "ba", "bb", and "bc"
Match any character in a range of characters	[a-f]	be[n-t] matches "bet" in "between", "ben" in "beneath", and "bes" in "beside", but finds no matches in "below"
Capture and implicitly number the expression contained within parenthesis	0	([a-z])X\1 matches "aXa" and "bXb", but not "aXb". "\1" refers to the first expression group "[a-z]". For more information, see Capture groups and replacement patterns .
Invalidate a match	(?!abc)	real(?!ity) matches "real" in "realty" and "really" but not in "reality." It also finds the second "real" (but not the first "real") in "realtyreal".
Match any character that is not in a given set of characters. For more information, see Negative character group .	[^abc]	be[^n-t] matches "bef" in "before", "beh" in "behind", and "bel" in "below", but finds no matches in "beneath"
Match either the expression before or the one after the symbol		(sponge mud) bath matches "sponge bath" and "mud bath"
Escape the character following the backslash	\	\^ matches the character ^
Specify the number of occurrences of the preceding character or group. For more information, see Match exactly n times .	{n}, where 'n' is the number of occurrences	x(ab){2}x matches "xababx" x(ab){2,3}x matches "xababx" and "xabababx" but not "xababababx"
Match text in a Unicode category. For more information about Unicode character classes, see Unicode Standard 5.2 Character Properties .	\p{X}, where "X" is the Unicode number.	\p{Lu} matches "T" and "D" in "Thomas Doe"
Match a word boundary	\b (Outside a character class \b specifies a word boundary, and inside a character class \b specifies a backspace.)	\bin matches "in" in "inside" but finds no matches in "pinto"
Match a line break (that is, a carriage return followed by a new line)	\r?\n	End\r?\nBegin matches "End" and "Begin" only when "End" is the last string in a line and "Begin" is the first string in the next line
Match any word character	\w	a\wd matches "add" and "a1d" but not "a d"

PURPOSE	EXPRESSION	EXAMPLE
Match any whitespace character	\s	Public\sInterface matches the phrase "Public Interface"
Match any decimal digit character	\d	\d matches "4" and "0" in "wd40"

An example regular expression that combines some of the operators and constructs to match a hexadecimal number is \b0[xx]([0-9a-fA-F]+)\b. This expression matches "0xc67f" but not "0xc67g".

TIP

In Windows operating systems, most lines end in "\r\n" (a carriage return followed by a new line). These characters aren't visible but are present in the editor and passed to the .NET regular expression service.

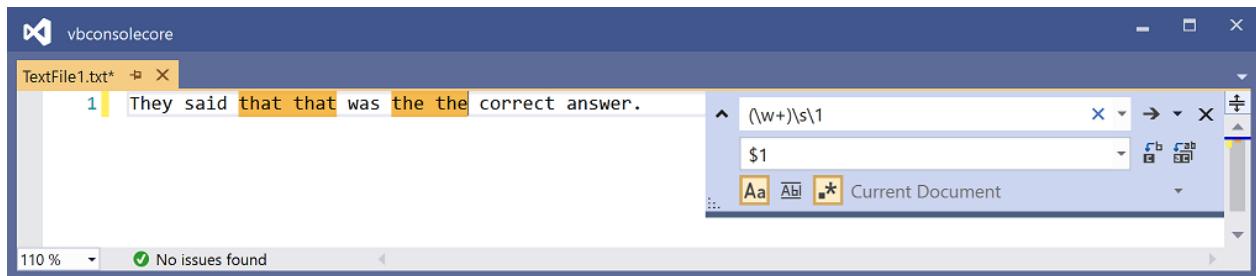
Capture groups and replacement patterns

A capture group delineates a subexpression of a regular expression and captures a substring of an input string. You can use captured groups within the regular expression itself (for example, to look for a repeated word), or in a replacement pattern. For detailed information, see [Grouping constructs in regular expressions](#).

To create a numbered capture group, surround the subexpression with parentheses in the regular expression pattern. Captures are numbered automatically from left to right based on the position of the opening parenthesis in the regular expression. To access the captured group:

- **within the regular expression:** Use \number. For example, \1 in the regular expression (\w+)\s\1 references the first capture group (\w+).
- **in a replacement pattern:** Use \$number. For example, the grouped regular expression (\d)([a-z]) defines two groups: the first group contains a single decimal digit, and the second group contains a single character between a and z. The expression finds four matches in the following string: 1a 2b 3c 4d. The replacement string z\$1 references the first group only (\$1), and converts the string to z1 z2 z3 z4.

The following image shows a regular expression (\w+)\s\1 and a replacement string \$1. Both the regular expression and the replacement pattern reference the first capture group that's automatically numbered 1. When you choose **Replace all** in the **Quick Replace** dialog box in Visual Studio, repeated words are removed from the text.



TIP

Make sure the **Use Regular Expressions** button is selected in the **Quick Replace** dialog box.

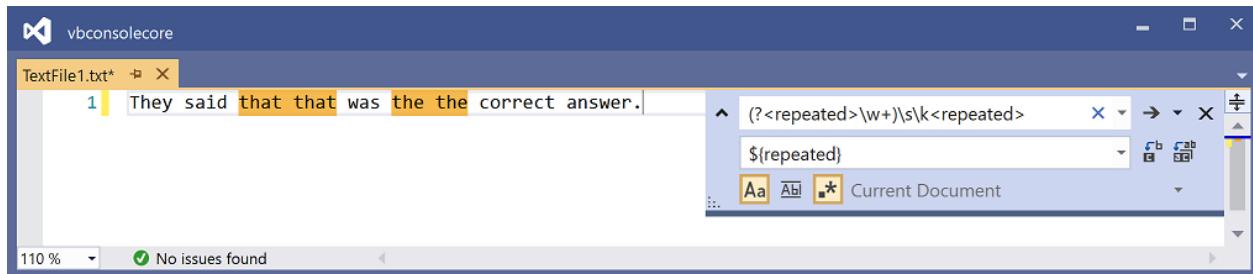
Named capture groups

Instead of relying on the automatic numbering of a capture group, you can give it a name. The syntax for a named capture group is (?<name>subexpression).

Named capture groups, like numbered capture groups, can be used within the regular expression itself or in a replacement pattern. To access the named capture group:

- **within the regular expression:** Use `\k<name>`. For example, `(?<repeated>\w+)\s\k<repeated>` in the regular expression references the capture group that's named `repeated` and whose subexpression is `\w+`.
- **in a replacement pattern:** Use `${name}`. For example, `${repeated}`.

As an example, the following image shows a regular expression `(?<repeated>\w+)\s\k<repeated>` and a replacement string `${repeated}`. Both the regular expression and the replacement pattern reference the capture group named `repeated`. When you choose **Replace all** in the **Quick Replace** dialog box in Visual Studio, repeated words are removed from the text.



TIP

Make sure the **Use Regular Expressions** button is selected in the **Quick Replace** dialog box.

For more information about named capture groups, see [Named matched subexpressions](#). For more information about regular expressions that are used in replacement patterns, see [Substitutions in regular expressions](#).

See also

- [Regular expression language](#)
- [Find and replace text](#)

Find/Command box

10/18/2019 • 2 minutes to read • [Edit Online](#)

You can search for text and run Visual Studio commands from the **Find/Command** box. The **Find/Command** box is still available as a toolbar control, but is no longer visible by default. You can display the **Find/Command** box by choosing **Add or Remove Buttons** on the **Standard** toolbar and then choosing **Find**.

To run a Visual Studio command, preface it with a greater than (>) sign.

The **Find/Command** box retains the last 20 items entered and displays them in a drop-down list. You can navigate through the list by choosing the **arrow keys**.



Searching for text

By default, when you specify text in the **Find/Command** box and then choose the **Enter** key, Visual Studio searches the current document or tool window using the options that are specified in the **Find in Files** dialog box. For more information, see [Finding and replacing text](#).

Entering commands

To use the **Find/Command** box to issue a single Visual Studio command or alias rather than search for text, preface the command with a greater than (>) symbol. For example:

```
>File.NewFile c:\temp\MyFile /t:"General\Text File"
```

Alternatively, you can also use the **Command** window to enter and execute single or multiple commands. Some commands or aliases can be entered and executed by themselves; others have required arguments in their syntax. For a list of commands that have arguments, see [Visual Studio commands](#).

Escape characters

A caret (^) character in a command means that the character immediately following it is interpreted literally, rather than as a control character. This can be used to embed straight quotation marks ("), spaces, leading slashes, carets, or any other literal characters in a parameter or switch value, with the exception of switch names. For example:

```
>Edit.Find ^^t /regex
```

A caret functions the same whether it is inside or outside quotation marks. If a caret is the last character on the line, it is ignored.

See also

- [Command window](#)
- [Finding and replacing text](#)

Find in Files

10/18/2019 • 3 minutes to read • [Edit Online](#)

Find in Files allows you to search a specified set of files. The matches found and actions taken are listed in the **Find Results** window selected in **Result options**.

You can use any of the following methods to display **Find in Files** in the **Find and Replace** window.

To display Find in Files

1. On the menu bar, choose **Edit > Find and Replace**.
2. Choose **Find in Files**.

To cancel a Find operation, press **Ctrl + Break**.

NOTE

The Find and Replace tool does not search directories with the `Hidden` or `System` attribute.

Find what

To search for a new text string or expression, specify it in the box. To search for any of the 20 strings that you searched for most recently, open the drop-down list and choose the string. Choose the adjacent **Expression Builder** button if you want to use one or more regular expressions in your search string. For more information, see [Using regular expressions in Visual Studio](#).

NOTE

The **Expression Builder** button will only be enabled if you have selected **Use Regular Expressions** under **Find options**.

Look in

The option chosen from the **Look in** drop-down list determines whether **Find in Files** searches only in currently active files or in all files stored within certain folders. Select a search scope from the list or click the **Browse (...)** button to display the **Choose Search Folders** dialog box and to enter your own set of directories. You can also type a path directly into the **Look in** box.

WARNING

With the **Entire Solution** or **Current Project** options, project and solution files are not searched. If you want to look in project files, choose a search folder.

NOTE

If the **Look in** option selected causes you to search a file that you have checked out from source code control, only the version of that file which has been downloaded to your local machine is searched.

Include subfolders

Specifies that subfolders of the **Look in** folder will be searched.

Find options

You can expand or collapse the **Find options** section. The following options can be selected or cleared:

Match case

When selected, a **Find Results** search will be case-sensitive

Match whole word

When selected, the **Find Results** windows will only return whole word matches.

Use Regular Expressions

If this check box is selected, you can use special notations to define patterns of text to match in the **Find what** or **Replace with** text boxes. For a list of these notations, see [Using regular expressions in Visual Studio](#).

Look at these file types

This list indicates the types of files to search through in the **Look in** directories. If this field is blank, all of the files in the **Look in** directories will be searched.

Select any item in the list to enter a preconfigured search string that will find files of those particular types.

Result options

You can expand or collapse the **Result options** section. The following options can be selected or cleared:

Find results 1 window

When selected, the results of the current search will replace the content of the **Find Results 1** window. This window opens automatically to display your search results. To open this window manually, select **Other Windows** from the **View** menu and choose **Find Results 1**.

Find results 2 window

When selected, the results of the current search will replace the content of the **Find Results 2** window. This window opens automatically to display your search results. To open this window manually, select **Other Windows** from the **View** menu and choose **Find Results 2**.

Display file names only

Displays a list of files containing search matches rather than displaying the search matches themselves.

Append results

Appends the results from the search to the previous search results.

See also

- [Finding and replacing text](#)
- [Replace in Files](#)
- [Visual Studio commands](#)

Replace in Files

10/18/2019 • 4 minutes to read • [Edit Online](#)

Replace in Files allows you to search the code of a specified set of files for a string or expression, and change some or all of the matches found. The matches found and actions taken are listed in the **Find Results** window selected in **Result options**.

NOTE

The dialog boxes and menu commands you see might differ from those described in **Help** depending on your active settings or edition. To change your settings, for example to **General** or **Visual C++** settings, choose **Tools > Import and Export Settings**, and then choose **Reset all settings**.

You can use any of the following methods to display **Replace in Files** in the **Find and Replace** window.

To display Replace in Files

1. On the **Edit** menu, expand **Find and Replace**.
2. Choose **Replace in Files**.

— or —

If the **Find and Replace** window is already open, on the toolbar, choose **Replace in Files**.

Find what

To search for a new text string or expression, specify it in the box. To search for any of the 20 strings that you searched for most recently, open the drop-down list and choose the string. Choose the adjacent **Expression Builder** button if you want to use one or more regular expressions in your search string. For more information, see [Use regular expressions in Visual Studio](#).

NOTE

The **Expression Builder** button will only be enabled if you have selected **Use Regular Expressions** under **Find options**.

Replace With

To replace instances of the string in the **Find what** box with another string, enter the replacement string in the **Replace With** box. To delete instances of the string in the **Find what** box, leave this field blank. Open the list to display the 20 strings for which you searched most recently. Choose the adjacent **Expression Builder** button if you want to use one or more regular expressions in your replacement string. For more information, see [Use regular expressions in Visual Studio](#).

Look in

The option chosen from the **Look in** drop-down list determines whether **Replace in Files** searches only in currently active files or searches all files stored within certain folders. Select a search scope from the list, type a folder path, or click the **Browse (...)** button to display the **Choose Search Folders** dialog box and choose a set of folders to search. You can also type a path directly into the **Look in** box.

NOTE

If the **Look in** option selected causes you to search a file that you have checked out from source code control, only the version of that file which has been downloaded to your local machine is searched.

Find options

You can expand or collapse the **Find options** section. The following options can be selected or cleared:

Match case

When selected, the **Find Results** windows will only display instances of the **Find what** string that are matched both by content and by case. For example, a search for "MyObject" with **Match case** selected will return "MyObject" but not "myobject" or "MYOBJECT."

Match whole word

When selected, the **Find Results** windows will only display instances of the **Find what** string that are matched in complete words. For example, a search for "MyObject" will return "MyObject" but not "CMyObject" or "MyObjectC."

Use Regular Expressions

When this check box is selected, you can use special notations to define patterns of text in the **Find what** or **Replace with** text boxes. For a list of these notations, see [Use regular expressions in Visual Studio](#).

Look at these file types

This list indicates the types of files to search through in the **Look in** directories. If this field is left blank, all of the files in the **Look in** directories will be searched. Select any item in the list to enter a preconfigured search string that will find files of those particular types.

Result options

You can expand or collapse the **Result options** section. The following options can be selected or cleared:

Find Results 1 window

When selected, the results of the current search will replace the content of the **Find Results 1** window. This window opens automatically to display your search results. To open this window manually, select **Other Windows** from the **View** menu and choose **Find Results 1**.

Find Results 2 window

When selected, the results of the current search will replace the content of the **Find Results 2** window. This window opens automatically to display your search results. To open this window manually, select **Other Windows** from the **View** menu and choose **Find Results 2**.

Display file names only

When this check box is selected, the **Find Results** windows list the full names and paths for all files that contain the search string. However, the results don't include the line of code where the string appears. This check box is available for **Find in Files** only.

Keep modified files open after Replace All

When selected, leaves open all files in which replacements have been made, so you can undo or save the changes. Memory constraints might limit the number of files that can remain open after a replace operation.

Caution

You can use **Undo** only on files that remain open for editing. If this option is not selected, files that were not already open for editing will remain closed, and no **Undo** option will be available in those files.

See also

- [Find and replace text](#)
- [Find in files](#)
- [Visual Studio commands](#)

Encodings and line endings

10/18/2019 • 2 minutes to read • [Edit Online](#)

The following characters are interpreted as line breaks in Visual Studio:

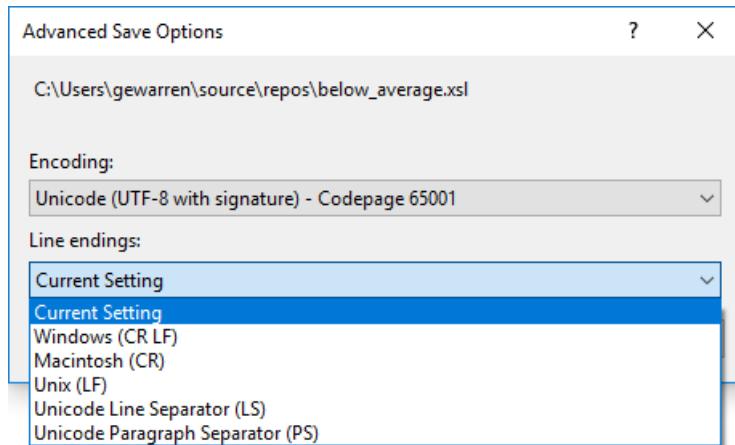
- CR LF: Carriage return + line feed, Unicode characters 000D + 000A
- LF: Line feed, Unicode character 000A
- NEL: Next line, Unicode character 0085
- LS: Line separator, Unicode character 2028
- PS: Paragraph separator, Unicode character 2029

Text that is copied from other applications keeps the original encoding and line break characters. For example, when you copy text from Notepad and paste it into a text file in Visual Studio, the text has the same settings that it had in Notepad.

When you open a file that has different line break characters, you may see a dialog box that asks whether the inconsistent line break characters should be normalized, and which type of line breaks to choose.

Advanced save options

You can use the **File > Advanced Save Options** dialog box to determine the type of line break characters you want. You can also change the encoding of a file with the same settings.



NOTE

If you don't see **Advanced Save Options** on the **File** menu, you can add it. Choose **Tools, Customize**, and then choose the **Commands** tab. In the **Menu bar** drop-down list, choose **File**, then choose the **Add Command** button. In the **Add Command** dialog box, under **Categories**, choose **File**, and then in the **Commands** list, choose **Advanced Save Options**. Choose **OK** and then choose the **Move Down** button to move the command to any place in the menu. Choose **Close** to close the **Customize** dialog box. For more information, see [Customize menus and toolbars](#).

Alternatively, you can access the **Advanced Save Options** dialog box by choosing **File > Save <file> As**. In the **Save File As** dialog box, choose the drop-down triangle next to the **Save** button and choose **Save with encoding**.

See also

- Features of the code editor

How to: Save and open files with encoding

10/18/2019 • 2 minutes to read • [Edit Online](#)

You can save files with specific character encoding to support bidirectional languages. You can also specify an encoding when opening a file, so that Visual Studio displays the file correctly.

To save a file with encoding

1. From the **File** menu, choose **Save File As**, and then click the drop-down button next to the **Save** button.

The **Advanced Save Options** dialog box is displayed.

2. Under **Encoding**, select the encoding to use for the file.
3. Optionally, under **Line endings**, select the format for end-of-line characters.

This option is useful if you intend to exchange the file with users of a different operating system.

If you want to work with a file that you know is encoded in a specific way, you can tell Visual Studio to use that encoding when opening the file. The method you use depends on whether the file is part of your project.

To open an encoded file that is part of a project

1. In **Solution Explorer**, right-click the file and choose **Open With**.
2. In the **Open With** dialog box, choose the editor to open the file with.

Many Visual Studio editors, such as the forms editor, will auto-detect the encoding and open the file appropriately. If you choose an editor that allows you to choose an encoding, the **Encoding** dialog box is displayed.

3. In the **Encoding** dialog box, select the encoding that the editor should use.

To open an encoded file that is not part of a project

1. On the **File** menu, point to **Open**, choose **File** or **File From Web**, and then select the file to open.
2. Click the drop-down button next to the **Open** button and choose **Open With**.
3. Follow Steps 2 and 3 from the preceding procedure.

See also

- [Encoding and line breaks](#)
- [Encoding and Windows Forms globalization](#)
- [Globalize and localize applications](#)

Outlining

10/21/2019 • 2 minutes to read • [Edit Online](#)

You can choose to hide some code from view by collapsing a region of code so that it appears under a plus sign (+). You expand a collapsed region by clicking the plus sign. If you are a keyboard user, you can choose **Ctrl+M+M** to collapse and expand. You can also collapse an outlining region by double-clicking any line in the region on the outlining margin, which appears just to the left of the code. You can see the contents of a collapsed region as a tooltip when you hover over the collapsed region.

NOTE

This topic applies to Visual Studio on Windows. For Visual Studio for Mac, see [Source editor \(Visual Studio for Mac\)](#).

Regions in the outlining margin are also highlighted when you hover over the margin with the mouse. The default highlighting color may seem rather faint in some color configurations. You can change it in **Tools > Options > Environment > Fonts and Colors > Collapsible Region**.

When you work in outlined code, you can expand the sections you want to work on, collapse them when you are done, and then move to other sections. When you do not wish to have outlining displayed, you can use the **Stop Outlining** command to remove the outline information without disturbing your underlying code.

The **Undo** and **Redo** commands on the **Edit** menu affect these actions. The **Copy**, **Cut**, **Paste**, and drag-and-drop operations retain outlining information, but not the state of the collapsible region. For example, when you copy a region that is collapsed, the **Paste** operation will paste the copied text as an expanded region.

Caution

When you change an outlined region, the outlining may be lost. For example, deletions or Find and Replace operations may erase the end of the region.

The following commands can be found on the **Edit > Outlining** submenu.

Hide Selection	(Ctrl+M, Ctrl+H) - Collapses a selected block of code that would not normally be available for outlining, for example an <code>if</code> block. To remove the custom region, use Stop Hiding Current (or Ctrl+M, Ctrl+U). Not available in Visual Basic.
Toggle Outlining Expansion	- Reverses the current hidden or expanded state of the innermost outlining section when the cursor lies in a nested collapsed section.
Toggle All Outlining	(Ctrl+M, Ctrl+L) - Sets all regions to the same collapsed or expanded state. If some regions are expanded and some collapsed, then the collapsed regions are expanded.
Stop Outlining	(Ctrl+M, Ctrl+P) - Removes all outlining information for the entire document.
Stop Hiding Current	(Ctrl+M, Ctrl+U) - Removes the outlining information for the currently selected user-defined region. Not available in Visual Basic.

Collapse to Definitions	(Ctrl+M, Ctrl+O) - Collapses the members of all types.
Collapse Block:<logical boundary>	(C++) Collapses a region in the function containing the insertion point. For example, if the insertion point lies inside a loop, the loop is hidden.
Collapse All in: <logical structures>	(C++) Collapses all the structures inside the function.

You can also use the Visual Studio SDK to define the text regions you want to expand or collapse. See [Walkthrough: Outlining](#).

See also

- [Features of the code editor](#)
- [Source editor \(Visual Studio for Mac\)](#)

Code generation features in Visual Studio

10/18/2019 • 2 minutes to read • [Edit Online](#)

There are numerous ways that Visual Studio can help you generate, fix, and refactor code.

Features

- You can use [code snippets](#) to insert a template such as a `switch` block or an `enum` declaration.
- You can use [Quick Actions](#) to generate code such as classes and properties, or to introduce a local variable. You can also use Quick Actions to [improve code](#), for example to remove unnecessary casts and unused variables, or to add null checks before accessing variables.
- You can [refactor code](#) to rename a variable, re-order method parameters, or synchronize a type with its filename, to name a few.

NOTE

Each language service in Visual Studio provides its own code generation capabilities, so some features are only available in C#, and some are available in both C# and Visual Basic.

See also

- [Code snippets](#)
- [Quick Actions](#)
- [Refactoring](#)
- [Code generation and T4 text templates](#)

Code snippets

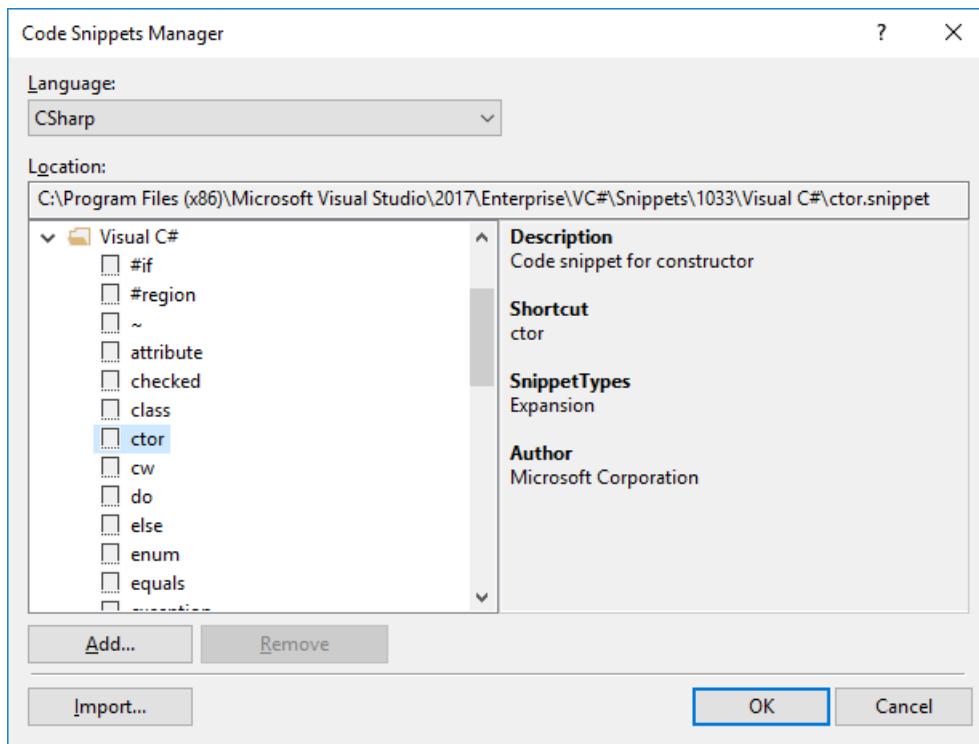
10/21/2019 • 2 minutes to read • [Edit Online](#)

Code snippets are small blocks of reusable code that can be inserted in a code file using a right-click menu (context menu) command or a combination of hotkeys. They typically contain commonly used code blocks such as `try-finally` or `if-else` blocks, but they can be used to insert entire classes or methods.

NOTE

This topic applies to Visual Studio on Windows. For Visual Studio for Mac, see [Code snippets \(Visual Studio for Mac\)](#).

Code snippets are available for a multitude of languages, including C#, C++, Visual Basic, XML, and T-SQL, to name a few. To view all the available installed snippets for a language, open the **Code Snippets Manager** from the **Tools** menu (or, press **Ctrl+K, Ctrl+B**), and choose the language from the drop-down menu at the top.



Code snippets can be accessed in the following general ways:

- On the menu bar, choose **Edit > IntelliSense > Insert Snippet**
- From the right-click or context menu in the code editor, choose **Snippet > Insert Snippet**
- From the keyboard, press **Ctrl+K,Ctrl+X**

Expansion snippets and surround-with snippets

In Visual Studio there are two kinds of code snippet: expansion snippets, which are added at a specified insertion point and may replace a snippet shortcut, and surround-with snippets (C# and C++ only), which are added around a selected block of code.

An example of an expansion snippet: in C# the shortcut `tryf` is used to insert a `try-finally` block:

```
try
{
}
finally
{
}
```

You can insert this snippet by clicking **Insert Snippet** in the right-click menu (context menu) of the code window, then **Visual C#**, then type `tryf`, and then press **Tab**. Or, you can type `tryf` and press **Tab** twice.

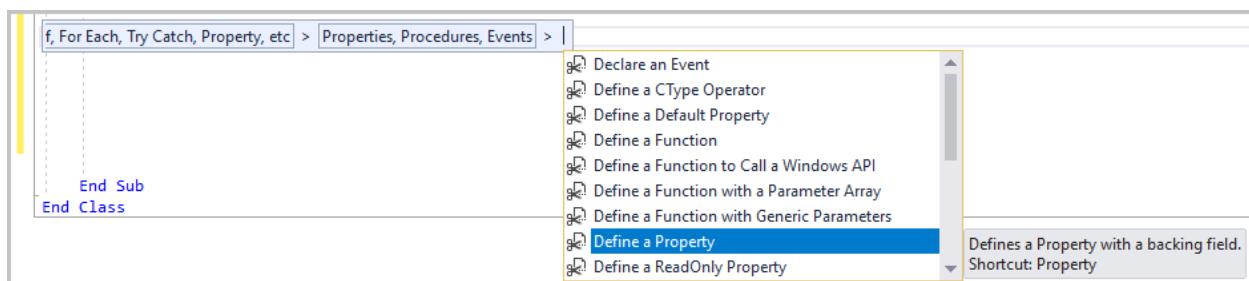
An example of a surround-with snippet: in C++ the shortcut `if` can be used either as an insertion snippet or as a surround-with snippet. If you select a line of code (for example `return FALSE;`), and then choose **Surround With** > **if**, the snippet is expanded around the line:

```
if (true)
{
    return FALSE;
}
```

Snippet replacement parameters

Snippets can contain replacement parameters, which are placeholders that you must replace to fit the precise code you are writing. In the previous example `true` is a replacement parameter, which you would replace with the appropriate condition. The replacement you make is repeated for every instance of the same replacement parameter in the snippet.

For example, in Visual Basic there's a code snippet that inserts a property. To insert the snippet, choose **Snippet** > **Insert Snippet** from the right-click or context menu in a Visual Basic code file. Then, choose **Code Patterns** > **Properties, Procedures, Events** > **Define a Property**.



The following code is inserted:

```
Private newPropertyValue As String
Public Property NewProperty() As String
    Get
        Return newPropertyValue
    End Get
    Set(ByVal value As String)
        newPropertyValue = value
    End Set
End Property
```

If you change `newPropertyValue` to `m_property`, then every instance of `newPropertyValue` is changed. If you change `String` to `Int` in the property declaration, then the value in the set method is also changed to `Int`.

See also

- [Walkthrough: Creating a code snippet](#)
- [How to: Distribute code snippets](#)
- [Best practices for using code snippets](#)
- [Troubleshooting snippets](#)
- [C# code snippets](#)
- [C++ code snippets](#)
- [Code snippets schema reference](#)
- [Code snippets \(Visual Studio for Mac\)](#)

C# code snippets

10/18/2019 • 4 minutes to read • [Edit Online](#)

Code snippets are ready-made snippets of code you can quickly insert into your code. For example, the `for` code snippet creates an empty `for` loop. Some code snippets are surround-with code snippets, which enable you to select lines of code, and then choose a code snippet which incorporates the selected lines of code. For example, when you select lines of code and then activate the `for` code snippet, it creates a `for` loop with those lines of code inside the loop block. Code snippets can make writing program code quicker, easier, and more reliable.

You can insert a code snippet at the cursor location, or insert a surround-with code snippet around the currently selected code. The Code Snippet Inserter is invoked through the **Insert Code Snippet** or **Surround With** commands on the **IntelliSense** menu, or by using the keyboard shortcuts **Ctrl+K,X** or **Ctrl+K,S** respectively.

The **Code Snippet Inserter** displays the code snippet name for all available code snippets. The Code Snippet Inserter also includes an input dialog box where you can type the name of the code snippet, or part of the code snippet name. The Code Snippet Inserter highlights the closest match to a code snippet name. Pressing **Tab** at any time will dismiss the Code Snippet Inserter and insert the currently selected code snippet. Pressing **Esc** or clicking the mouse in the code editor will dismiss the Code Snippet Inserter without inserting a code snippet.

Default code snippets

By default the following code snippets are included in Visual Studio for C#.

NAME (OR SHORTCUT)	DESCRIPTION	VALID LOCATIONS TO INSERT SNIPPET
#if	Creates a <code>#if</code> directive and a <code>#endif</code> directive.	Anywhere.
#region	Creates a <code>#region</code> directive and a <code>#endregion</code> directive.	Anywhere.
~	Creates a <code>finalizer</code> (destructor) for the containing class.	Inside a class.
attribute	Creates a declaration for a class that derives from Attribute .	Inside a namespace (including the global namespace), a class, or a struct.
checked	Creates a <code>checked</code> block.	Inside a method, an indexer, a property accessor, or an event accessor.
class	Creates a class declaration.	Inside a namespace (including the global namespace), a class, or a struct.
ctor	Creates a constructor for the containing class.	Inside a class.
cw	Creates a call to WriteLine .	Inside a method, an indexer, a property accessor, or an event accessor.
do	Creates a <code>do</code> <code>while</code> loop.	Inside a method, an indexer, a property accessor, or an event accessor.

NAME (OR SHORTCUT)	DESCRIPTION	VALID LOCATIONS TO INSERT SNIPPET
else	Creates an <code>else</code> block.	Inside a method, an indexer, a property accessor, or an event accessor.
enum	Creates an <code>enum</code> declaration.	Inside a namespace (including the global namespace), a class, or a struct.
equals	Creates a method declaration that overrides the <code>Equals</code> method defined in the <code>Object</code> class.	Inside a class or a struct.
exception	Creates a declaration for a class that derives from an exception (<code>Exception</code> by default).	Inside a namespace (including the global namespace), a class, or a struct.
for	Creates a <code>for</code> loop.	Inside a method, an indexer, a property accessor, or an event accessor.
foreach	Creates a <code>foreach</code> loop.	Inside a method, an indexer, a property accessor, or an event accessor.
forr	Creates a <code>for</code> loop that decrements the loop variable after each iteration.	Inside a method, an indexer, a property accessor, or an event accessor.
if	Creates an <code>if</code> block.	Inside a method, an indexer, a property accessor, or an event accessor.
indexer	Creates an indexer declaration.	Inside a class or a struct.
interface	Creates an <code>interface</code> declaration.	Inside a namespace (including the global namespace), a class, or a struct.
invoke	Creates a block that safely invokes an event.	Inside a method, an indexer, a property accessor, or an event accessor.
iterator	Creates an iterator.	Inside a class or a struct.
iterindex	Creates a "named" iterator and indexer pair by using a nested class.	Inside a class or a struct.
lock	Creates a <code>lock</code> block.	Inside a method, an indexer, a property accessor, or an event accessor.
mbox	Creates a call to <code>System.Windows.Forms.MessageBox.Show</code> . You may have to add a reference to <code>System.Windows.Forms.dll</code> .	Inside a method, an indexer, a property accessor, or an event accessor.
namespace	Creates a <code>namespace</code> declaration.	Inside a namespace (including the global namespace).
prop	Creates an <code>auto-implemented property</code> declaration.	Inside a class or a struct.

NAME (OR SHORTCUT)	DESCRIPTION	VALID LOCATIONS TO INSERT SNIPPET
propfull	Creates a property declaration with <code>get</code> and <code>set</code> accessors.	Inside a class or a struct.
propg	Creates a read-only auto-implemented property with a private <code>set</code> accessor.	Inside a class or a struct.
sim	Creates a <code>static int Main</code> method declaration.	Inside a class or a struct.
struct	Creates a <code>struct</code> declaration.	Inside a namespace (including the global namespace), a class, or a struct.
svm	Creates a <code>static void Main</code> method declaration.	Inside a class or a struct.
switch	Creates a <code>switch</code> block.	Inside a method, an indexer, a property accessor, or an event accessor.
try	Creates a <code>try-catch</code> block.	Inside a method, an indexer, a property accessor, or an event accessor.
tryf	Creates a <code>try-finally</code> block.	Inside a method, an indexer, a property accessor, or an event accessor.
unchecked	Creates an <code>unchecked</code> block.	Inside a method, an indexer, a property accessor, or an event accessor.
unsafe	Creates an <code>unsafe</code> block.	Inside a method, an indexer, a property accessor, or an event accessor.
using	Creates a <code>using</code> directive.	Inside a namespace (including the global namespace).
while	Creates a <code>while</code> loop.	Inside a method, an indexer, a property accessor, or an event accessor.

See also

- [Code snippet functions](#)
- [Code snippets](#)
- [Template parameters](#)
- [How to: Use surround-with code snippets](#)

Visual C++ code snippets

7/24/2019 • 2 minutes to read • [Edit Online](#)

In Visual Studio, you can use code snippets to add commonly-used code to your C++ code files. In general, you can use code snippets in much the same way as in C#, but the set of default code snippets is different.

You can either add a code snippet at a particular location in your code (insertion) or surround some selected code with a code snippet.

Insert a code snippet

To insert a code snippet, open a C++ code file (.cpp or .h), click somewhere inside the file, and do one of the following:

- Right-click to get the context menu and select **Insert Snippet**
- In the **Edit / IntelliSense** menu, select **Insert Snippet**
- Use the hotkeys: **Ctrl+K+X**

You should see a list of choices beginning with **#if**. When you select **#if**, you should see the following code added to the file:

```
#if 0  
#endif // 0
```

You can then replace the **0** with the correct condition.

Use a code snippet to surround selected code

To use a code snippet to surround selected code, select a line (or multiple lines) and do one of the following:

- Right-click to get the context menu, and select **Surround With**
- From the **Edit > IntelliSense** menu, select **Surround With**
- Using a keyboard, press: **Ctrl+K+S**

Select **#if**. You should see something like this:

```
#if 0  
#include "pch.h" // or whatever line you had selected  
#endif // 0
```

You can then replace the **0** with the correct condition.

Where can I find a complete list of the C++ code snippets?

You can find the complete list of C++ code snippets by going to the **Code Snippets Manager** (on the **Tools** menu) and setting the **Language** to **Visual C++**. In the window below, expand **Visual C++**. You should see the names of all the C++ code snippets in alphabetical order.

The names of most code snippets are self-explanatory, but some names might be confusing.

Class vs. classi

The **class** snippet provides the definition of a class named `MyClass`, with the appropriate default constructor and destructor, where the definitions of the constructor and destructor are located outside the class:

```
class MyClass
{
public:
    MyClass();
    ~MyClass();

private:
};

MyClass::MyClass()
{
}

MyClass::~MyClass()
{
}
```

The **classi** code snippet also provides the definition of a class named `MyClass`, but the default constructor and destructor are defined inside the class definition:

```
class MyClass
{
public:
    MyClass()
    {

    ~MyClass()
    {

private:
};
```

for vs. forr vs rfor

There are three different **for** snippets that provide different kinds of `for` loops.

The **rfor** snippet provides a [range-based](#) for loop (link). This construct is preferred over index-based `for` loops.

```
for (auto& i : v)
{
```

The **for** snippet provides a `for` loop in which the condition is based on the length (in `size_t`) of an object.

```
for (size_t i = 0; i < length; i++)  
{  
}
```

The **forr** snippet provides a reverse `for` loop in which the condition is based on the length (in integers) of an object.

```
for (int i = length - 1; i >= 0; i--)  
{  
}
```

The destructor snippet (~)

The destructor snippet (~) shows different behavior in different contexts. If you insert this snippet inside a class, it provides a destructor for that class. For example, given the following code:

```
class SomeClass {  
};
```

If you insert the destructor snippet, it provides a destructor for `SomeClass`:

```
class SomeClass {  
    ~SomeClass()  
    {  
    }  
};
```

If you try to insert the destructor snippet outside a class, it provides a destructor with a placeholder name:

```
~TypeNamePlaceholder()  
{
```

See also

- [Code snippets](#)

How to: Insert XML comments for documentation generation

10/18/2019 • 2 minutes to read • [Edit Online](#)

Visual Studio can help you document code elements such as classes and methods, by automatically generating the standard XML documentation comment structure. At compile time, you can generate an XML file that contains the documentation comments.

TIP

For information about configuring the name and location of the generated XML file, see [Documenting your code with XML comments \(C# Guide\)](#).

The compiler-generated XML file can be distributed alongside your .NET assembly so that Visual Studio and other IDEs can use IntelliSense to show quick information about types and members. Additionally, the XML file can be run through tools like [DocFX](#) and [Sandcastle](#) to generate API reference websites.

NOTE

The **Insert Comment** command that automatically inserts XML documentation comments is available in [C#](#) and [Visual Basic](#). However, you can manually insert [XML documentation comments in C++ files](#) and still generate XML documentation files at compile time.

To insert XML comments for a code element

1. Place your text cursor above the element you want to document, for example, a method.
2. Do one of the following:
 - Type `///` in C#, or `'''` in Visual Basic
 - From the **Edit** menu, choose **IntelliSense > Insert Comment**
 - From the right-click or context menu on or just above the code element, choose **Snippet > Insert Comment**

The XML template is immediately generated above the code element. For example, when commenting a method, it generates the `<summary>` element, a `<param>` element for each parameter, and a `<returns>` element to document the return value.

```
///<summary>
///|
///</summary>
///<param name="id"></param>
///<returns></returns>
public string GetUsername(int id)
{
    return "username";
}
```

```
''' <summary>
''' 
''' </summary>
''' <param name="id"></param>
''' <returns></returns>
Public Function GetUsername(id As Integer) As String
    Return "username"
End Function
```

3. Enter descriptions for each XML element to fully document the code element.

```
/// <summary>
/// Gets the username associated with the specified ID.
/// </summary>
/// <param name="id">The unique user ID.</param>
/// <returns>A string containing the username for the ID specified.</returns>
public string GetUsername(int id)
{
    return "username";
}
```

NOTE

There is an [option](#) to toggle XML documentation comments after typing `///` in C# or `'''` Visual Basic. From the menu bar, choose **Tools** > **Options** to open the **Options** dialog box. Then, navigate to **Text Editor** > **C#** or **Basic** > **Advanced**. In the **Editor Help** section, look for the **Generate XML documentation comments** option.

See also

- [XML documentation comments \(C# Programming Guide\)](#)
- [Documenting your code with XML comments \(C# Guide\)](#)
- [How to: Create XML documentation \(Visual Basic\)](#)
- [C++ Comments](#)
- [XML Documentation \(C++\)](#)
- [Code generation](#)

How to: Use surround-with code snippets

10/18/2019 • 2 minutes to read • [Edit Online](#)

The following procedures describe how to use surround-with code snippets. Surround-with code snippets are available three ways: through a keyboard shortcut, through the **Edit** menu, and through the right-click or context menu.

To use surround-with code snippets through keyboard shortcut

1. In the Visual Studio IDE, open the file that you intend to edit.
2. In the **Code Editor**, select text to surround.
3. Type **Ctrl+K, Ctrl+S**.
4. Select the code snippet from the code snippet list using the mouse, or by typing the name of the code snippet and pressing **Tab** or **Enter**.

To use surround-with code snippets through the Edit menu

1. In the Visual Studio IDE, open the file that you intend to edit.
2. In the **Code Editor**, select text to surround.
3. From the **Edit** menu, select **IntelliSense** and then select the **Surround With** command.
4. Select the code snippet from the code snippet inserter and then press **Tab** or **Enter**.

Alternatively, you can type the name of the code snippet, and then press **Tab** or **Enter**.

To use surround-with code snippets through the context menu

1. In the Visual Studio IDE, open the file that you intend to edit.
2. In the **Code Editor**, select text to surround.
3. Right-click the selected text and then select the **Surround With** command from the context menu.
4. Select the code snippet from the code snippet inserter and then press **Tab** or **Enter**.

Alternatively, you can type the name of the code snippet, and then press **Tab** or **Enter**.

See also

- [C# code snippets](#)
- [Code snippet picker](#)

Best practices for using code snippets

10/18/2019 • 2 minutes to read • [Edit Online](#)

The code in a code snippet shows only the most basic way to do something. For most applications, the code must be modified to suit the application.

Handling exceptions

Typically, code snippet Try...Catch blocks catch and rethrow all exceptions. That may not be the right choice for your project. For each exception, there are several ways to respond. For examples, see [How to: Handle an exception using try/catch \(C#\)](#) and [Try...Catch...Finally statement \(Visual Basic\)](#).

File locations

When you adapt file locations to your application, you should think about the following:

- Finding an accessible location. Users may not have access to the *Program Files* folder of the computer, so storing files with the application files may not work.
- Finding a secure location. Storing files in the root folder (C:\) is not secure. For application data, we recommend the *Application Data* folder. For individual user data, the application can create a file for each user in the *Documents* folder.
- Using a valid file name. You can use the [OpenFileDialog](#) and [SaveFileDialog](#) controls to reduce the likelihood of invalid file names. Be aware that between the time the user selects a file and the time your code manipulates the file, the file may be deleted. In addition, the user may not have permissions to write to the file.

Security

How secure a snippet is depends on where it is used in the source code and how it is modified once it is in the code. The following list contains a few of the areas that must be considered.

- File and database access
- Code access security
- Protecting resources (such as event logs, registry)
- Storing secrets
- Verifying inputs
- Passing data to scripting technologies

For more information, see [Securing applications](#).

Downloaded code snippets

IntelliSense code snippets installed by Visual Studio are not in themselves a security hazard. However, they can create security risks in your application. Snippets downloaded from the Internet should be treated like any other downloaded content - with extreme caution.

- Download snippets only from sites you trust, and use up-to-date virus software.

- Open all downloaded snippet files in Notepad or the XML editor of Visual Studio and review them carefully before installing them. Look for the following issues:
 - The snippet code could damage your system if you execute it. Read the source code carefully before running it.
 - The Help URL block of the snippet file can contain URLs that execute a malicious script file or display an offensive website.
 - The snippet may contain references that are added silently to your project and may be loaded from anywhere on your system. These references may have been downloaded to your computer from where you downloaded the snippet. The snippet may then make a call to a method in the reference that executes malicious code. To protect yourself against such an attack, review the Imports and References blocks of the snippet file.

See also

- [Visual Basic IntelliSense code snippets](#)
- [Securing applications](#)
- [Code snippets](#)

Walkthrough: Create a code snippet

10/18/2019 • 5 minutes to read • [Edit Online](#)

You can create a code snippet with only a few steps. All you need to do is create an XML file, fill in the appropriate elements, and add your code to it. You can optionally make use of replacement parameters and project references. Import the snippet to your Visual Studio installation by using the **Import** button in the **Code Snippets Manager** (**Tools > Code Snippets Manager**).

Snippet template

The following XML is the basic snippet template:

```
<?xml version="1.0" encoding="utf-8"?>
<CodeSnippets xmlns="http://schemas.microsoft.com/VisualStudio/2005/CodeSnippet">
    <CodeSnippet Format="1.0.0">
        <Header>
            <Title></Title>
        </Header>
        <Snippet>
            <Code Language="">
                <![CDATA[]]>
            </Code>
        </Snippet>
    </CodeSnippet>
</CodeSnippets>
```

Create a code snippet

1. Create a new XML file in Visual Studio and add the template shown above.
2. Fill in the title of the snippet in the **Title** element. Use the title **Square Root**.
3. Fill in the language of the snippet in the **Language** attribute of the **Code** element. For C#, use **CSharp**, and for Visual Basic, use **VB**.

TIP

To see all the available language values, browse the [Code element attributes section](#) on the [Code snippets schema reference](#) page.

4. Add the snippet code in the **CDATA** section inside the **Code** element.

For C#:

```
<Code Language="CSharp">
    <![CDATA[Double root = Math.Sqrt(16);]>
</Code>
```

Or for Visual Basic:

```
<Code Language="VB">
<![CDATA[Dim root = Math.Sqrt(16)]]>
</Code>
```

NOTE

You can't specify how lines of code in the **CDATA** section of a code snippet should be indented or formatted. Upon insertion, the language service formats the inserted code automatically.

5. Save the snippet as *SquareRoot.snippet* (you can save it anywhere).

Import a code snippet

1. You can import a snippet to your Visual Studio installation by using the **Code Snippets Manager**. Open it by choosing **Tools > Code Snippets Manager**.
2. Click the **Import** button.
3. Go to the location where you saved the code snippet in the previous procedure, select it, and click **Open**.
4. The **Import Code Snippet** dialog opens, asking you to choose where to add the snippet from the choices in the right pane. One of the choices should be **My Code Snippets**. Select it and click **Finish**, then **OK**.
5. The snippet is copied to one of the following locations, depending on the code language:

%USERPROFILE%\Documents\Visual Studio 2017\Code Snippets\Visual C#\My Code Snippets
%USERPROFILE%\Documents\Visual Studio 2017\Code Snippets\Visual Basic\My Code Snippets

%USERPROFILE%\Documents\Visual Studio 2019\Code Snippets\Visual C#\My Code Snippets
%USERPROFILE%\Documents\Visual Studio 2019\Code Snippets\Visual Basic\My Code Snippets

6. Test your snippet by opening a C# or Visual Basic project. With a code file open in the editor, choose **Snippets > Insert Snippet** from the right-click menu, then **My Code Snippets**. You should see a snippet named **Square Root**. Double-click it.

The snippet code is inserted in the code file.

Description and shortcut fields

1. Description fields give more information about your code snippet when viewed in the Code Snippets Manager. The shortcut is a tag that users can type in order to insert your snippet. Edit the snippet you have added by opening the file %USERPROFILE%\Documents\Visual Studio 2017\Code Snippets\[Visual C# or Visual Basic]\My Code Snippet\SquareRoot.snippet.
1. Description fields give more information about your code snippet when viewed in the Code Snippets Manager. The shortcut is a tag that users can type in order to insert your snippet. Edit the snippet you have added by opening the file %USERPROFILE%\Documents\Visual Studio 2019\Code Snippets\[Visual C# or Visual Basic]\My Code Snippet\SquareRoot.snippet.

TIP

Since you're editing the file in the directory where Visual Studio placed it, you don't need to reimport it to Visual Studio.

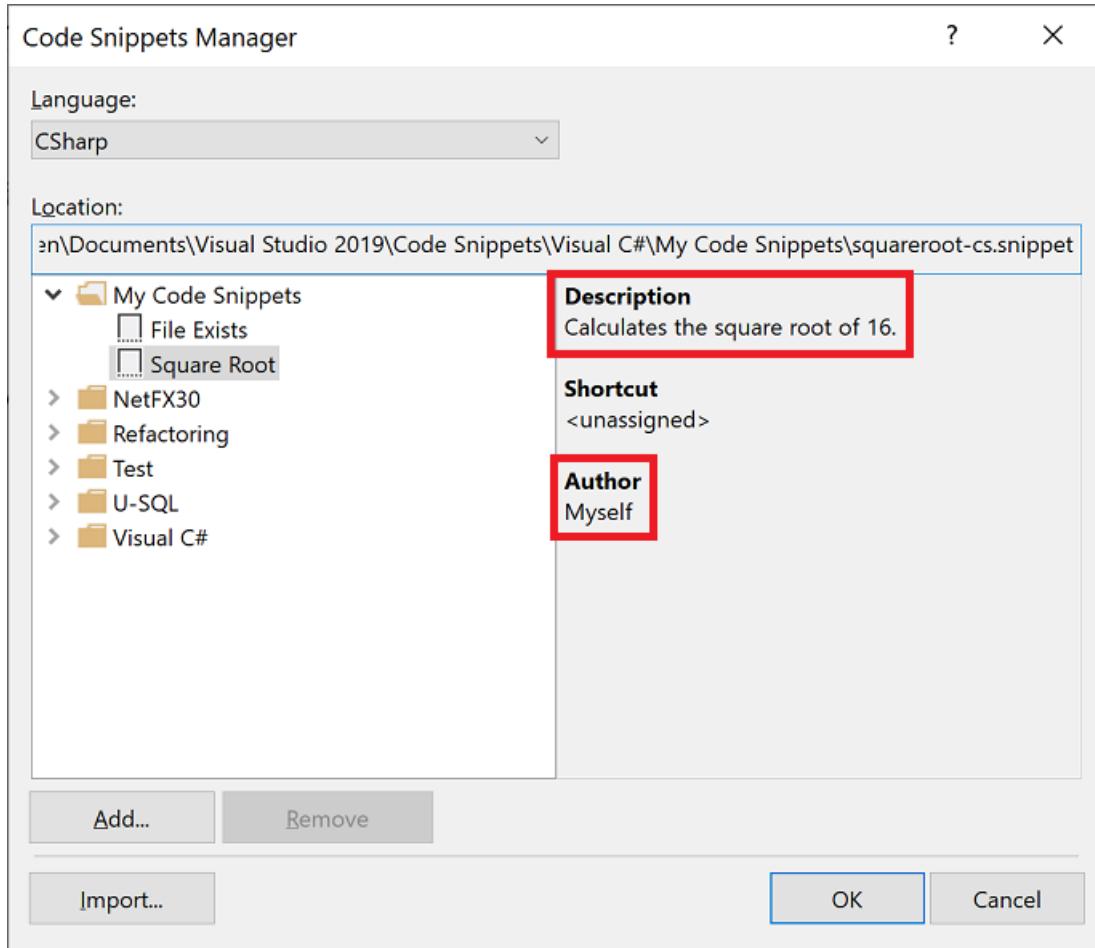
2. Add **Author** and **Description** elements to the **Header** element, and fill them in.
3. The **Header** element should look something like this:

```

<Header>
  <Title>Square Root</Title>
  <Author>Myself</Author>
  <Description>Calculates the square root of 16.</Description>
</Header>

```

4. Open the **Code Snippets Manager** and select your code snippet. In the right pane, notice that the **Description** and **Author** fields are now populated.



5. To add a shortcut, add a **Shortcut** element within the **Header** element:

```

<Header>
  <Title>Square Root</Title>
  <Author>Myself</Author>
  <Description>Calculates the square root of 16.</Description>
  <Shortcut>sqrt</Shortcut>
</Header>

```

6. Save the snippet file again.

7. To test the shortcut, open the project you used previously, type **sqrt** in the editor and press **Tab** (once for Visual Basic, twice for C#).

The snippet code is inserted.

Replacement parameters

You may want parts of a code snippet to be replaced by the user. For example, you might want the user to replace a variable name with one in their current project. You can provide two types of replacements: literals and objects.

Use the [Literal element](#) to identify a replacement for a piece of code that is entirely contained within the snippet but will likely be customized after it's inserted into the code (for example, a string or numeric value). Use the [Object element](#) to identify an item that's required by the code snippet but is likely to be defined outside of the snippet itself (for example, an object instance or a control).

1. To enable the user to easily replace the number to calculate the square root of, modify the **Snippet** element of the *SquareRoot.snippet* file as follows:

```
<Snippet>
<Code Language="CSharp">
<![CDATA[double root = Math.Sqrt($Number$);]]>
</Code>
<Declarations>
<Literal>
<ID>Number</ID>
<ToolTip>Choose the number you want the square root of.</ToolTip>
<Default>16</Default>
</Literal>
</Declarations>
</Snippet>
```

Notice that the literal replacement is given an ID (`$Number$`). That ID is referenced from within the code snippet by surrounding it with `$` characters:

```
<![CDATA[double root = Math.Sqrt($Number$);]]>
```

2. Save the snippet file.
3. Open a project and insert the snippet.

The code snippet is inserted and the editable literal is highlighted for replacement. Hover over the replacement parameter to see the tooltip for the value.

A screenshot of a code editor showing a tooltip for a placeholder in a snippet. The code is `double root = Math.Sqrt(16);`. A tooltip box appears over the number `16`, containing the text: **struct System.Int32** Represents a 32-bit signed integer. Below the tooltip, a red box highlights the placeholder `$Number$` in the original code, with the tooltip text Choose the number you want the square root of. above it.

TIP

If there's more than one replacable parameter in a snippet, you can press **Tab** to navigate from one to the other to change the values.

Import a namespace

You can use a code snippet to add a `using` directive (C#) or `Imports` statement (Visual Basic) by including the [Imports element](#). For .NET Framework projects, you can also add a reference to the project by using the [References element](#).

The following XML shows a code snippet that uses the method `File.Exists` in the `System.IO` namespace and, therefore, defines the **Imports** element to import the `System.IO` namespace.

```
<?xml version="1.0" encoding="utf-8"?>
<CodeSnippets xmlns="http://schemas.microsoft.com/VisualStudio/2005/CodeSnippet">
  <CodeSnippet Format="1.0.0">
    <Header>
      <Title>File Exists</Title>
      <Shortcut>exists</Shortcut>
    </Header>
    <Snippet>
      <Code Language="CSharp">
        <![CDATA[var exists = File.Exists("C:\\Temp\\\\Notes.txt");]]>
      </Code>
      <Imports>
        <Import>
          <Namespace>System.IO</Namespace>
        </Import>
      </Imports>
    </Snippet>
  </CodeSnippet>
</CodeSnippets>
```

See also

- [Code snippets schema reference](#)

How to: Distribute code snippets

10/18/2019 • 3 minutes to read • [Edit Online](#)

You can give your code snippets to your friends and have them install the snippets on their own computers by using **Code Snippets Manager**. However, if you have several snippets to distribute or would like to distribute them more widely, you can include your snippet files in a Visual Studio extension. Visual Studio users can then install the extension to obtain the snippets.

Prerequisites

Install the **Visual Studio extension development** workload to get access to the **VSIX Project** project templates.



Set up the extension

In this procedure, you'll use the same Hello World code snippet that's created in [Walkthrough: Create a code snippet](#). This article provides the snippet XML, so you don't have to go back and create a snippet.

1. Create a new project from the **Empty VSIX Project** template and name the project **TestSnippet**.
2. In the **TestSnippet** project, add a new XML file and call it *VBCodeSnippet.snippet*. Replace the content with the following XML:

```
<?xml version="1.0" encoding="utf-8"?>
<CodeSnippets
    xmlns="http://schemas.microsoft.com/VisualStudio/2005/CodeSnippet">
    <CodeSnippet Format="1.0.0">
        <Header>
            <Title>Hello World VB</Title>
            <Shortcut>HelloWorld</Shortcut>
            <Description>Inserts code</Description>
            <Author>MSIT</Author>
            <SnippetTypes>
                <SnippetType>Expansion</SnippetType>
                <SnippetType>SurroundsWith</SnippetType>
            </SnippetTypes>
        </Header>
        <Snippet>
            <Code Language="VB">
                <![CDATA[Console.WriteLine("Hello, World!")]]>
            </Code>
        </Snippet>
    </CodeSnippet>
</CodeSnippets>
```

Set up the directory structure

1. In **Solution Explorer**, select the project node and add a folder that has the name you want the snippet to have in **Code Snippets Manager**. In this case, it should be **HelloWorldVB**.
2. Move the *.snippet* file to the *HelloWorldVB* folder.

3. Select the *.snippet* file in **Solution Explorer**, and in the **Properties** window make sure **Build Action** is set to **Content**, **Copy to Output Directory** is set to **Copy always**, and **Include in VSIX** is set to **true**.

Add the *.pkgdef* file

1. Add a text file to the *HelloWorldVB* folder and name it *HelloWorldVB.pkgdef*. This file is used to add certain keys to the registry. In this case, it adds a new subkey to the **HKEY_CURRENT_USER\Software\Microsoft\VisualStudio\15.0\Languages\CodeExpansions\Basic** key.
1. Add a text file to the *HelloWorldVB* folder and name it *HelloWorldVB.pkgdef*. This file is used to add certain keys to the registry. In this case, it adds a new subkey to the **HKEY_CURRENT_USER\Software\Microsoft\VisualStudio\16.0\Languages\CodeExpansions\Basic** key.
2. Add the following lines to the file.

```
// Visual Basic
[$RootKey$\Languages\CodeExpansions\Basic\Paths]
"HelloWorldVB"="$PackageFolder$"
```

If you examine this key, you can see how to specify different languages.

3. Select the *.pkgdef* file in **Solution Explorer**, and in the **Properties** window make sure that:
 - **Build Action** is set to **Content**
 - **Copy to Output Directory** is set to **Copy always**
 - **Include in VSIX** is set to **true**
4. Add the *.pkgdef* file as an asset in the VSIX manifest. In the *source.extension.vsixmanifest* file, go to the **Assets** tab and click **New**.
5. In the **Add New Asset** dialog, set the **Type** to **Microsoft.VisualStudio.VsPackage**, the **Source** to **File on filesystem**, and the **Path** to **HelloWorldVB.pkgdef** (which should appear in the dropdown).

Test the snippet

1. Now you can make sure that the code snippet works in the experimental instance of Visual Studio. The experimental instance is a second copy of Visual Studio that is separate from the one you use to write code. It allows you to work on an extension without affecting your development environment.
2. Build the project and start debugging.

A second instance of Visual Studio appears.

3. In the experimental instance, go to **Tools > Code Snippets Manager** and set the **Language** to **Basic**. You should see *HelloWorldVB* as one of the folders, and you should be able to expand the folder to see the *HelloWorldVB* snippet.
4. Test the snippet. In the experimental instance, open a Visual Basic project and open one of the code files. Place your cursor somewhere in the code, right-click, and on the context menu select **Insert Snippet**.
5. You should see *HelloWorldVB* as one of the folders. Double-click it. You should see a pop-up **Insert Snippet: HelloWorldVB** > that has a dropdown **HelloWorldVB**. Click the **HelloWorldVB** dropdown.

The following line is added to the code file:

```
Console.WriteLine("Hello, World!")
```

See also

- [Code snippets](#)

Code snippet functions

10/18/2019 • 2 minutes to read • [Edit Online](#)

There are three functions available to use with C# code snippets. Functions are specified in the `Function` element of the code snippet. For information on creating code snippets, see [Code snippets](#).

Functions

The following table describes the functions available for use with the `Function` element in code snippets.

FUNCTION	DESCRIPTION	LANGUAGE
<code>GenerateSwitchCases(EnumerationLiteral)</code>	Generates a switch statement and a set of case statements for the members of the enumeration specified by the <code>EnumerationLiteral</code> parameter. The <code>EnumerationLiteral</code> parameter must be either a reference to an enumeration literal or an enumeration type.	C#
<code>ClassName()</code>	Returns the name of the class that contains the inserted snippet.	C#
<code>SimpleTypeName(TypeName)</code>	Reduces the <code>TypeName</code> parameter to its simplest form in the context in which the snippet was invoked.	C#

GenerateSwitchCases example

The following example shows how to use the `GenerateSwitchCases` function. When this snippet is inserted and an enumeration is entered into the `$switch_on$` literal, the `$cases$` literal generates a `case` statement for every value in the enumeration.

```

<CodeSnippets xmlns="http://schemas.microsoft.com/VisualStudio/2005/CodeSnippet">
  <CodeSnippet Format="1.0.0">
    <Header>
      <Title>switch</Title>
      <Shortcut>switch</Shortcut>
      <Description>Code snippet for switch statement</Description>
      <Author>Microsoft Corporation</Author>
      <SnippetTypes>
        <SnippetType>Expansion</SnippetType>
      </SnippetTypes>
    </Header>
    <Snippet>
      <Declarations>
        <Literal>
          <ID>expression</ID>
          <ToolTip>Expression to switch on</ToolTip>
          <Default>switch_on</Default>
        </Literal>
        <Literal Editable="false">
          <ID>cases</ID>
          <Function>GenerateSwitchCases($expression$)</Function>
          <Default>default:</Default>
        </Literal>
      </Declarations>
      <Code Language="csharp">
        <![CDATA[
          switch ($expression$)
          {
            $cases$
          }
        ]]>
      </Code>
    </Snippet>
  </CodeSnippet>
</CodeSnippets>

```

ClassName example

The following example shows how to use the `className` function. When this snippet is inserted, the `$classname$` literal is replaced with the name of the enclosing class at that location in the code file.

```

<CodeSnippets xmlns="http://schemas.microsoft.com/VisualStudio/2005/CodeSnippet">
  <CodeSnippet Format="1.0.0">
    <Header>
      <Title>Common constructor pattern</Title>
      <Shortcut>ctor</Shortcut>
      <Description>Code Snippet for a constructor</Description>
      <Author>Microsoft Corporation</Author>
      <SnippetTypes>
        <SnippetType>Expansion</SnippetType>
      </SnippetTypes>
    </Header>
    <Snippet>
      <Declarations>
        <Literal>
          <ID>type</ID>
          <Default>int</Default>
        </Literal>
        <Literal>
          <ID>name</ID>
          <Default>field</Default>
        </Literal>
        <Literal default="true" Editable="false">
          <ID>classname</ID>
          <ToolTip>Class name</ToolTip>
          <Function>ClassName()</Function>
          <Default>ClassNamePlaceholder</Default>
        </Literal>
      </Declarations>
      <Code Language="csharp" Format="CData">
        <![CDATA[
          public $classname$ ($type$ $name$)
          {
            this._$name$ = $name$;
          }
          private $type$ _$name$;
        ]]>
      </Code>
    </Snippet>
  </CodeSnippet>
</CodeSnippets>

```

SimpleTypeName example

This example shows how to use the `simpleTypeName` function. When this snippet is inserted into a code file, the `$SystemConsole$` literal will be replaced with the simplest form of the `Console` type in the context in which the snippet was invoked.

```
<CodeSnippets xmlns="http://schemas.microsoft.com/VisualStudio/2005/CodeSnippet">
  <CodeSnippet Format="1.0.0">
    <Header>
      <Title>Console_WriteLine</Title>
      <Shortcut>cw</Shortcut>
      <Description>Code snippet for Console.WriteLine</Description>
      <Author>Microsoft Corporation</Author>
      <SnippetTypes>
        <SnippetType>Expansion</SnippetType>
      </SnippetTypes>
    </Header>
    <Snippet>
      <Declarations>
        <Literal Editable="false">
          <ID>SystemConsole</ID>
          <Function>SimpleTypeName(global::System.Console)</Function>
        </Literal>
      </Declarations>
      <Code Language="csharp">
        <![CDATA[
          $SystemConsole$.WriteLine();
        ]]>
      </Code>
    </Snippet>
  </CodeSnippet>
</CodeSnippets>
```

See also

- [Function element](#)
- [Code snippets schema reference](#)

Code snippets schema reference

10/18/2019 • 16 minutes to read • [Edit Online](#)

IntelliSense Code Snippets are pre-authored pieces of code that are ready to be inserted into your application with Visual Studio. You can increase productivity by providing code snippets that reduce the amount of time spent typing repetitive code or searching for samples. You can use the IntelliSense Code Snippet XML schema to create your own code snippets and add them to the code snippets that Visual Studio already includes.

Assembly element

Specifies the name of the assembly referenced by the code snippet.

The text value of the **Assembly** element is either the friendly text name of the assembly, such as `System.dll`, or its strong name, such as `System,Version=1.0.0.1,Culture=neutral,PublicKeyToken=9b35aa323c18d4fb1`.

```
<Assembly>
  AssemblyName
</Assembly>
```

PARENT ELEMENT	DESCRIPTION
Reference element	Contains information about assembly references required by the code snippet.

A text value is required. This text specifies the assembly that the code snippet references.

Author element

Specifies the name of the snippet author. The **Code Snippets Manager** displays the name stored in the `Author` element of the code snippet.

```
<Author>
  Code Snippet Author
</Author>
```

PARENT ELEMENT	DESCRIPTION
Header element	Contains general information about the code snippet.

A text value is required. This text specifies the author of the code snippet.

Code element

Provides a container for short code blocks.

Keywords

Two reserved words are available for use in the text of the `Code` element: `end` and `$selected$`. `end` marks the location to place the cursor after the code snippet is inserted. `$selected$` represents text selected in the document that is to be inserted into the snippet when it is invoked. For example, given a snippet that includes:

```
$selected$ is a great color.
```

If the word "Blue" is selected when the user invokes the template, the result is:

```
Blue is a great color.
```

You may not use either `end` or `$selected$` more than once in a code snippet. If you do, only the second instance is recognized. Given a snippet that includes:

```
$selected$ is a great color. I love $selected$.
```

If the word "Blue" is selected, the result is:

```
is a great color. I love Blue.
```

The initial space appears because there is a space between `$selected$` and `is`.

All other `$` keywords are dynamically defined in the `<Literal>` and `<Object>` tags.

Following is the structure of the Code element:

```
<Code Language="Language"
      Kind="method body/method decl/type decl/page/file/any"
      Delimiter="Delimiter">
  Code to insert
</Code>
```

A text value is required. This text specifies the code, along with the literals and objects, that you can use when this code snippet is inserted into a code file.

Attributes

There are three attributes available for the Code element:

- **Language** - *Required* attribute that specifies the language of the code snippet. The value can be one of the following:

VALUE	DESCRIPTION
VB	Identifies a Visual Basic code snippet.
CSharp	Identifies a C# code snippet.
CPP	Identifies a C++ code snippet.
XML	Identifies an XML code snippet.
JavaScript	Identifies a JavaScript code snippet.
TypeScript	Identifies a TypeScript code snippet.
SQL	Identifies a SQL code snippet.

VALUE	DESCRIPTION
HTML	Identifies an HTML code snippet.

- **Kind** - *Optional* attribute that specifies the kind of code that the snippet contains. The value can be one of the following:

VALUE	DESCRIPTION
method body	Specifies that the code snippet is a method body, and therefore, must be inserted inside a method declaration.
method decl	Specifies that the code snippet is a method, and therefore, must be inserted inside a class or module.
type decl	Specifies that the code snippet is a type, and therefore, must be inserted inside a class, module, or namespace.
file	Specifies that the snippet is a full code file. These code snippets can be inserted alone into a code file, or inside a namespace.
any	Specifies that the snippet can be inserted anywhere. This tag is used for code snippets that are context-independent, such as comments.

- **Delimiter** - *Optional* attribute that specifies the delimiter used to describe literals and objects in the code. By default, the delimiter is `$`.

Parent element

PARENT ELEMENT	DESCRIPTION
Snippet element	Contains the references, imports, declarations, and code for the code snippet.

CodeSnippet element

Allows you to specify a heading and multiple IntelliSense Code Snippets, which you can insert into Visual Studio code files.

```
<CodeSnippet Format="x.x.x">
  <Header>... </Header>
  <Snippet>... </Snippet>
</CodeSnippet>
```

ATTRIBUTE	DESCRIPTION
Format	Required attribute. Specifies the schema version of the code snippet. The Format attribute must be a string in the syntax of x.x.x, where each "x" represents a numerical value of the version number. Visual Studio will ignore code snippets with <code>Format</code> attributes that it does not understand.

CHILD ELEMENT	DESCRIPTION
Header element	Required element. Contains general information about the code snippet. There must be exactly one <code>Header</code> element in a code snippet.
Snippet element	Required element. Contains the code that will be inserted by Visual Studio. There must be exactly one <code>Snippet</code> element in a code snippet.
PARENT ELEMENT	DESCRIPTION
CodeSnippets element	Root element of the code snippet XML schema.

CodeSnippets element

Groups `CodeSnippet` elements. The `CodeSnippets` element is the root element of the code snippet XML schema.

```
<CodeSnippets>
  <CodeSnippet>... </CodeSnippet>
</CodeSnippets>
```

CHILD ELEMENT	DESCRIPTION
CodeSnippet element	Optional element. Parent element for all code snippet data. There may be zero or more <code>CodeSnippet</code> elements in a <code>CodeSnippets</code> element.

Declarations element

Specifies the literals and objects that make up the parts of a code snippet that you can edit.

```
<Declarations>
  <Literal>... </Literal>
  <Object>... </Object>
</Declarations>
```

CHILD ELEMENT	DESCRIPTION
Literal element	Optional element. Defines the literals of the code snippet that you can edit. There may be zero or more <code>Literal</code> elements in a <code>Declarations</code> element.
Object element	Optional element. Defines the objects of the code snippet that you can edit. There may be zero or more <code>Object</code> elements in a <code>Declarations</code> element.
PARENT ELEMENT	DESCRIPTION
Snippet element	Contains the references, imports, declarations, and code for the code snippet.

Default element

Specifies the default value of the literal or object for an IntelliSense Code Snippet.

```
<Default>
  Default value
</Default>
```

PARENT ELEMENT	DESCRIPTION
Literal element	Defines the literal fields of the code snippet that you can edit.
Object element	Defines the object fields of the code snippet that you can edit.

A text value is required. This text specifies the default value of the literal or object that fills the fields of the code snippet that you can edit.

Description element

Specifies descriptive information about the contents of an IntelliSense Code Snippet.

```
<Description>
  Code Snippet Description
</Description>
```

PARENT ELEMENT	DESCRIPTION
Header element	Contains general information about the code snippet.

A text value is required. This text describes the code snippet.

Function element

Specifies a function to execute when the literal or object receives focus in Visual Studio.

NOTE

The `Function` element is only supported in C# code snippets.

```
<Function>
  FunctionName
</Function>
```

PARENT ELEMENT	DESCRIPTION
Literal element	Defines the literal fields of the code snippet that you can edit.
Object element	Defines the object fields of the code snippet that you can edit.

A text value is required. This text specifies a function to execute when the literal or object field receives focus in Visual Studio.

Header element

Specifies general information about the IntelliSense Code Snippet.

```
<Header>
  <Title>... </Title>
  <Author>... </Author>
  <Description>... </Description>
  <HelpUrl>... </HelpUrl>
  <SnippetTypes>... </SnippetTypes>
  <Keywords>... </Keywords>
  <Shortcut>... </Shortcut>
</Header>
```

CHILD ELEMENT	DESCRIPTION
Author element	Optional element. The name of the person or company that authored the code snippet. There may be zero or one Author elements in a Header element.
Description element	Optional element. A description of the code snippet. There may be zero or one Description elements in a Header element.
HelpUrl element	Optional element. A URL that contains more information about the code snippet. There may be zero or one HelpURL elements in a Header element. Note: Visual Studio does not use the HelpUrl element. The element is part of the IntelliSense Code Snippet XML schema and any code snippet containing the element will validate, but the value of the element is never used.
Keywords element	Optional element. Groups Keyword elements. There may be zero or one Keywords elements in a Header element.
Shortcut element	Optional element. Specifies the shortcut text that can be used to insert the snippet. There may be zero or one Shortcut elements in a Header element.
SnippetTypes element	Optional element. Groups SnippetType elements. There may be zero or one SnippetTypes elements in a Header element. If there are no SnippetTypes elements, the code snippet is always valid.
Title element	Required element. The friendly name of the code snippet. There must be exactly one Title element in a Header element.
PARENT ELEMENT	DESCRIPTION
CodeSnippet element	Parent element for all code snippet data.

HelpUrl element

Specifies a URL that provides more information about a code snippet.

NOTE

Visual Studio does not use the `HelpUrl` element. The element is part of the IntelliSense Code Snippet XML schema and any code snippet containing the element will validate, but the value of the element is never used.

```
<HelpUrl>
  www.microsoft.com
</HelpUrl>
```

PARENT ELEMENT	DESCRIPTION
Header element	Contains general information about the code snippet.

A text value is optional. This text specifies the URL to visit for more information about a code snippet.

ID element

Specifies a unique identifier for a `Literal` or `Object` element. No two literals or objects in the same code snippet can have the same text value in their `ID` elements. Literals and objects cannot contain an `ID` element with a value of end. The value `end` is reserved, and is used to mark the location to place the cursor after the code snippet is inserted.

```
<ID>
  Unique Identifier
</ID>
```

PARENT ELEMENT	DESCRIPTION
Literal element	Defines the literal fields of the code snippet that you can edit.
Object element	Defines the object fields of the code snippet that you can edit.

A text value is required. This text specifies the unique identifier for the object or literal.

Import element

Specifies the imported namespaces used by an IntelliSense code snippet.

```
<Import>
  <Namespace>... </Namespace>
</Import>
```

CHILD ELEMENT	DESCRIPTION
Namespace element	Required element. Specifies the namespace used by the code snippet. There must be exactly one <code>Namespace</code> element in an <code>Import</code> element.

PARENT ELEMENT	DESCRIPTION
Imports element	Grouping element for Import elements.

Imports element

Groups individual **Import** elements.

```
<Imports>
  <Import>... </Import>
</Imports>
```

CHILD ELEMENT	DESCRIPTION
Import element	Optional element. Contains the imported namespaces for the code snippet. There may be zero or more Import elements in an Imports element.
PARENT ELEMENT	DESCRIPTION
Snippet element	Contains the references, imports, declarations, and code for the code snippet.

Keyword element

Specifies a custom keyword for the code snippet. The code snippet keywords are used by Visual Studio and represent a standard way for online content providers to add custom keywords for searching or categorization.

```
<Keyword>
  Code Snippet Keyword
</Keyword>
```

PARENT ELEMENT	DESCRIPTION
Keywords element	Groups individual Keyword elements.

A text value is required. The keyword for the code snippet.

Keywords element

Groups individual **Keyword** elements. The code snippet keywords are used by Visual Studio and represent a standard way for online content providers to add custom keywords for searching or categorization

```
<Keywords>
  <Keyword>... </Keyword>
  <Keyword>... </Keyword>
</Keywords>
```

CHILD ELEMENT	DESCRIPTION

CHILD ELEMENT	DESCRIPTION
Keyword element	Optional element. Contains individual keywords for the code snippet. There may be zero or more <code>Keyword</code> elements in a <code>Keywords</code> element.
PARENT ELEMENT	DESCRIPTION
Header element	Contains general information about the code snippet.

Literal element

Defines the literals of the code snippet that you can edit. The `Literal` element is used to identify a replacement for a piece of code that is entirely contained within the snippet, but will likely be customized after it is inserted into the code. For example, literal strings, numeric values, and some variable names should be declared as literals.

Literals and objects cannot contain an `ID` element with a value of selected or end. The value `$selected$` represents text selected in the document that is to be inserted into the snippet when it is invoked. `end` marks the location to place the cursor after the code snippet is inserted.

```
<Literal Editable="true/false">
  <ID>... </ID>
  <ToolTip>... </ToolTip>
  <Default>... </Default>
  <Function>... </Function>
</Literal>
```

ATTRIBUTE	DESCRIPTION
<code>Editable</code>	Optional <code>Boolean</code> attribute. Specifies whether or not you can edit the literal after the code snippet is inserted. The default value of this attribute is <code>true</code> .
CHILD ELEMENT	DESCRIPTION
Default element	Required element. Specifies the literal's default value when you insert the code snippet. There must be exactly one <code>Default</code> element in a <code>Literal</code> element.
Function element	Optional element. Specifies a function to execute when the literal receives focus in Visual Studio. There may be zero or one <code>Function</code> elements in a <code>Literal</code> element.
ID element	Required element. Specifies a unique identifier for the literal. There must be exactly one <code>ID</code> element in a <code>Literal</code> element.
ToolTip element	Optional element. Describes the expected value and usage of the literal. There may be zero or one <code>Tooltip</code> elements in a <code>Literal</code> element.

PARENT ELEMENT	DESCRIPTION
Declarations element	Contains the literals and objects of a code snippet that you can edit.

Namespace element

Specifies the namespace that must be imported for the code snippet to compile and run. The namespace specified in the `Namespace` element is automatically added to a `using` directive or `Imports` statement at the beginning of the code if it doesn't already exist.

```
<Namespace>
  Namespace
</Namespace>
```

PARENT ELEMENT	DESCRIPTION
Import element	Imports the specified namespace.

A text value is required. This text specifies a namespace that the code snippet assumes is imported.

Object element

Defines the objects of the code snippet that you can edit. The `Object` element is used to identify an item that is required by the code snippet but is likely to be defined outside of the snippet itself. For example, Windows Forms controls, ASP.NET controls, object instances, and type instances should be declared as objects. Object declarations require that a type be specified, which is done with the `Type` element.

```
<Object Editable="true/false">
  <ID>... </ID>
  <Type>... </Type>
  <ToolTip>... </ToolTip>
  <Default>... </Default>
  <Function>... </Function>
</Object>
```

ATTRIBUTE	DESCRIPTION
<code>Editable</code>	Optional <code>Boolean</code> attribute. Specifies whether or not you can edit the literal after the code snippet is inserted. The default value of this attribute is <code>true</code> .
CHILD ELEMENT	DESCRIPTION
Default element	Required element. Specifies the literal's default value when you insert the code snippet. There must be exactly one <code>Default</code> element in a <code>Literal</code> element.
Function element	Optional element. Specifies a function to execute when the literal receives focus in Visual Studio. There may be zero or one <code>Function</code> elements in a <code>Literal</code> element.

CHILD ELEMENT	DESCRIPTION
ID element	Required element. Specifies a unique identifier for the literal. There must be exactly one <code>ID</code> element in a <code>Literal</code> element.
ToolTip element	Optional element. Describes the expected value and usage of the literal. There may be zero or one <code>Tooltip</code> elements in a <code>Literal</code> element.
Type element	Required element. Specifies the type of the object. There must be exactly one <code>Type</code> element in an <code>Object</code> element.

PARENT ELEMENT	DESCRIPTION
Declarations element	Contains the literals and objects of a code snippet that you can edit.

Reference element

Specifies information about the assembly references required by the code snippet.

```
<Reference>
  <Assembly>... </Assembly>
  <Url>... </Url>
</Reference>
```

CHILD ELEMENT	DESCRIPTION
Assembly element	Required element. Contains the name of the assembly referenced by the code snippet. There must be exactly one <code>Assembly</code> element in a <code>Reference</code> element.
Url element	Optional element. Contains a URL that provides more information about the referenced assembly. There may be zero or one <code>Url</code> elements in a <code>Reference</code> element.
PARENT ELEMENT	DESCRIPTION
References element	Grouping element for <code>Reference</code> elements.

References element

Groups individual `Reference` elements.

```
<References>
  <Reference>... </Reference>
</References>
```

CHILD ELEMENT	DESCRIPTION

CHILD ELEMENT	DESCRIPTION
Reference element	Optional element. Contains information about assembly references for the code snippet. There may be zero or more Reference elements in a References element.
PARENT ELEMENT	DESCRIPTION
Snippet element	Contains the references, imports, declarations, and code for the code snippet.

Shortcut element

Specifies the shortcut text used to insert the snippet. The text value of a Shortcut element can only contain alphanumeric characters, hyphens (-), and underscores (_).

Caution

_ and - are not supported characters in C++ snippet shortcuts.

```
<Shortcut>
    Shortcut Text
</Shortcut>
```

PARENT ELEMENT	DESCRIPTION
Header element	Contains general information about the code snippet.

A text value is optional. This text is used as a shortcut for inserting the code snippet.

Snippet element

Specifies the references, imports, declarations, and code for the code snippet.

```
<Snippet>
    <References>... </References>
    <Imports>... </Imports>
    <Declarations>... </Declarations>
    <Code>... </Code>
</Snippet>
```

CHILD ELEMENT	DESCRIPTION
Code element	Required element. Specifies the code that you want to insert into a documentation file. There must be exactly one Code element in a Snippet element.
Declarations element	Optional element. Specifies the literals and objects that make up the parts of a code snippet that you can edit. There may be zero or one Declarations elements in a Snippet element.
Imports element	Optional element. Groups individual Import elements. There may be zero or one Imports elements in a Snippet element.

CHILD ELEMENT	DESCRIPTION
References element	Optional element. Groups individual Reference elements. There may be zero or one References elements in a Snippet element.
PARENT ELEMENT	DESCRIPTION
CodeSnippet element	Allows you to specify a heading and multiple IntelliSense Code Snippets, which you can insert into Visual Studio code files.

SnippetType element

Specifies how Visual Studio inserts the code snippet.

<pre><SnippetType> SurroundsWith/Expansion </SnippetType></pre>	
PARENT ELEMENT	DESCRIPTION
SnippetTypes element	Groups SnippetType elements.

The text value must be one of the following values:

- SurroundsWith : allows the code snippet to be placed around a selected piece of code.
- Expansion : allows the code snippet to be inserted at the cursor.
- Refactoring : specifies that the code snippet is used during C# refactoring. Refactoring cannot be used in custom code snippets.

SnippetTypes element

Groups individual SnippetType elements. If the SnippetTypes element is not present, the code snippet can be inserted anywhere in the code.

<pre><SnippetTypes> <SnippetType>... </SnippetType> <SnippetType>... </SnippetType> </SnippetTypes></pre>	
CHILD ELEMENT	DESCRIPTION
SnippetType element	Optional element. Specifies how Visual Studio inserts the code snippet into the code. There may be zero or more SnippetType elements in a SnippetTypes element.
PARENT ELEMENT	DESCRIPTION
Header element	Specifies general information about the code snippet.

Title element

Specifies the title for the code snippet. The title stored in the `Title` element of the code snippet appears in the **Code Snippet Picker** and in the code snippet's description in the **Code Snippets Manager**.

```
<Title>
  Code Snippet Title
</Title>
```

PARENT ELEMENT	DESCRIPTION
Header element	Specifies general information about the code snippet.

A text value is required. This text specifies the title of the code snippet.

ToolTip element

Describes the expected value and usage of a literal or object in a code snippet, which Visual Studio displays in a ToolTip when it inserts the code snippet into a project. The ToolTip text is displayed when the mouse hovers over the literal or object after the code snippet has been inserted.

```
<ToolTip>
  ToolTip description
</ToolTip>
```

PARENT ELEMENT	DESCRIPTION
Literal element	Defines the literal fields of the code snippet that you can edit.
Object element	Defines the object fields of the code snippet that you can edit.

A text value is required. This text specifies the ToolTip description to be associated with the object or literal in the code snippet.

Type element

Specifies the type of the object. The `Object` element is used to identify an item that is required by the code snippet but is likely to be defined outside of the snippet itself. For example, Windows Forms controls, ASP.NET controls, object instances, and type instances should be declared as objects. Object declarations require that a type be specified, which is done with the `Type` element.

```
<Type>
  Type
</Type>
```

PARENT ELEMENT	DESCRIPTION
Object element	Defines the object fields of the code snippet that you can edit.

A text value is required. This text specifies the type of the object. For example:

```
<Type>System.Data.SqlClient.SqlConnection</Type>
```

Url element

Specifies a URL that provides more information about the referenced assembly.

NOTE

The `Url` element is only supported for Visual Basic projects.

```
<Url>
  www.microsoft.com
</Url>
```

PARENT ELEMENT	DESCRIPTION
Reference element	Specifies the assembly references required by the code snippet.

A text value is required. This text specifies a URL with more information about the referenced assembly. This URL is displayed when the reference cannot be added to the project.

See also

- [Code snippets](#)
- [Walkthrough: Create a code snippet](#)

Troubleshoot snippets

10/18/2019 • 2 minutes to read • [Edit Online](#)

Problems with IntelliSense code snippets are typically caused by two problems: a corrupt snippet file or bad content in the snippet file.

The snippet cannot be dragged from File Explorer to a Visual Studio source file

- The XML in the snippet file may be corrupt. The **XML Editor** in Visual Studio can locate problems in the XML structure.
- The snippet file may not conform to the snippet schema. The **XML Editor** in Visual Studio can locate problems in the XML structure.

The code has compiler errors that are not highlighted

- You may be missing a project reference. Examine the documentation about the snippet. If the reference is not found on the computer, you will need to install it. Inserting a snippet should add to the project any references needed. If the snippet is missing the reference information, that can be reported to the snippet creator as an error.
- A variable may be undefined. Undefined variables in a snippet should be highlighted. If not, that can be reported to the snippet creator as an error.

See also

- [Code snippets](#)

Quick Actions

10/18/2019 • 2 minutes to read • [Edit Online](#)

Quick Actions let you easily refactor, generate, or otherwise modify code with a single action. Quick Actions are available for C#, C++, and Visual Basic code files. Some actions are specific to a language, and others apply to all languages.

Quick Actions can be used to:

- Apply a code fix for a [code analyzer](#) rule violation
- [Suppress](#) a code analyzer rule violation or [configure](#) its severity
- [Suppress](#) a code analyzer rule violation
- Apply a refactoring (for example, [inline a temporary variable](#))
- Generate code (for example, [introduce a local variable](#))

NOTE

This topic applies to Visual Studio on Windows. For Visual Studio for Mac, see [Refactoring \(Visual Studio for Mac\)](#).

Quick Actions can be applied by using the light bulb or screwdriver icons, or by pressing **Ctrl+.** when your cursor is on a line of code for which an action is available. You'll see an error light bulb if there's a red squiggle indicating an error and Visual Studio has a fix available for that error.

For any language, third parties can provide custom diagnostics and suggestions, for example as part of an SDK, and Visual Studio light bulbs appear based on those rules.

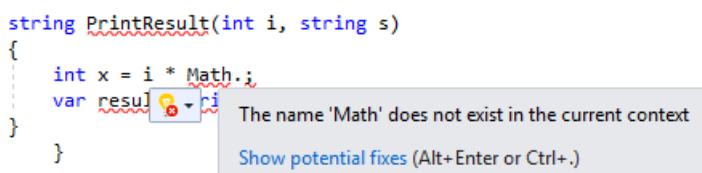
Icons

The icon that appears when a Quick Action is available gives an indication of the type of fix or refactoring that's available. The *screwdriver* icon indicates just that there are actions available to change the code, but you shouldn't necessarily use them. The *yellow light bulb* icon indicates there are actions available that you *should* do to improve your code. The *error light bulb* icon indicates there's an action available that fixes an error in your code.

To see a light bulb or screwdriver

If a fix is available, light bulbs appear:

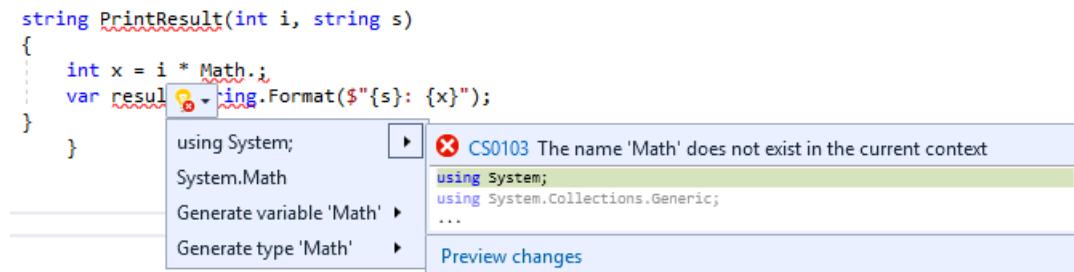
- When you hover the mouse at the location of an error



- In the left margin of the editor when you move the caret (cursor) into the applicable line of code

You can also press **Ctrl+.** anywhere on a line to see a list of available Quick Actions and refactorings.

To see potential fixes, select either the down arrow next to the light bulb or the **Show potential fixes** link. A list of available Quick Actions is displayed.



See also

- [Code generation in Visual Studio](#)
- [Common Quick Actions](#)
- [Code styles and Quick Actions](#)
- [Write and refactor code \(C++\)](#)
- [Refactoring \(Visual Studio for Mac\)](#)

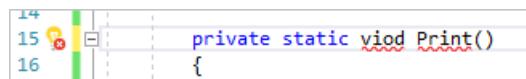
Common Quick Actions

10/18/2019 • 13 minutes to read • [Edit Online](#)

The sections in this topic list some of the common **Quick Actions** that are applicable to both C# and Visual Basic code. These actions are *code fixes* for either compiler diagnostics, or the built-in [.NET Compiler Platform analyzers](#) in Visual Studio.

Actions that fix errors

The Quick Actions in this section fix errors in code that would cause a build to fail. When Quick Actions are available to fix an error on a line of code, the icon that's displayed in the margin or underneath the red squiggle is a light bulb with a red 'x' on it.



Correct misspelled symbol or keyword

If you accidentally misspell a type or keyword in Visual Studio, this Quick Action automatically corrects it for you. You'll see these items in the light bulb menu as "**Change '<misspelled word>' to '<correct word>'**". For example:

```
// Before
private viod MyMethod()
{
}

// Change 'viod' to 'void'

// After
private void MyMethod()
{}
```

```
' Before
Function MyFunction as Intger
End Function

' Change 'Intger' to 'Integer'

' After
Function MyFunction as Integer
End Function
```

ERROR ID	APPLICABLE LANGUAGES
CS0103, BC30002	C# and Visual Basic

Resolve git merge conflict

These Quick Actions enable you to resolve git merge conflicts by "taking a change", which removes the conflicting code and markers.

```

// Before
private void MyMethod()
{
    if (false)
    {

    }
}

// Take changes from 'HEAD'

// After
private void MyMethod()
{
    if (true)
    {

    }
}

```

ERROR ID	APPLICABLE LANGUAGES	SUPPORTED VERSION
CS8300, BC37284	C# and Visual Basic	Visual Studio 2017 version 15.3 and later

Actions that remove unnecessary code

Remove unnecessary usings/Imports

The **Remove Unnecessary Usings/Imports** Quick Action removes any unused `using` and `Import` directives for the current file. When you select this item, unused namespace imports are removed.

APPLICABLE LANGUAGES	SUPPORTED VERSION
C# and Visual Basic	Visual Studio 2015 and later

Remove unnecessary cast

If you cast a type to another type that doesn't require a cast, the **Remove Unnecessary Cast** Quick Action item removes the unnecessary cast.

```

// before
int number = (int)3;

// Remove Unnecessary Cast

// after
int number = 3;

```

```

' Before
Dim number as Integer = CType(3, Integer)

' Remove Unnecessary Cast

' After
Dim number as Integer = 3

```

DIAGNOSTIC ID	APPLICABLE LANGUAGES	SUPPORTED VERSION
IDE0004	C# and Visual Basic	Visual Studio 2015 and later

Remove unused variables

This Quick Action enables you to remove variables that have been declared but never used in your code.

```
// Before
public MyMethod()
{
    var unused = 8;
    var used = 1;
    return DoStuff(used);
}

// Remove unused variables

// After
public MyMethod()
{
    var used = 1;
    return DoStuff(used);
}
```

DIAGNOSTIC ID	APPLICABLE LANGUAGES	SUPPORTED VERSION
CS0219, BC42024	C# and Visual Basic	Visual Studio 2017 version 15.3 and later

Remove type from default value expression

This Quick Action removes the value type from a default value expression and uses the [default literal](#) when the compiler can infer the type of the expression.

```
// Before
void DoWork(CancellationToken cancellationToken = default(CancellationToken)) { ... }

// Simplify default expression

// After
void DoWork(CancellationToken cancellationToken = default) { ... }
```

DIAGNOSTIC ID	APPLICABLE LANGUAGES	SUPPORTED VERSION
IDE0034	C# 7.1+	Visual Studio 2017 version 15.3 and later

Actions that add missing code

Add usings/imports for types in reference assemblies, NuGet packages, or other types in your solution

Using types located in other projects in your solution will display the Quick Action automatically, however the others need to be enabled from the **Tools > Options > C# or Basic > Advanced** tab:

- Suggest usings/imports for types in reference assemblies
- Suggest usings/imports for types in NuGet packages

When enabled, if you use a type in a namespace that is currently not imported but exists in a reference assembly or NuGet package, the using or import directive is created.

```
// Before  
Debug.WriteLine("Hello");  
  
// using System.Diagnostics;  
  
// After  
using System.Diagnostics;  
  
Debug.WriteLine("Hello");
```

```
' Before  
Debug.WriteLine("Hello")  
  
' Imports System.Diagnostics  
  
' After  
Imports System.Diagnostics  
  
Debug.WriteLine("Hello")
```

DIAGNOSTIC ID	APPLICABLE LANGUAGES
CS0103, BC30451	C# and Visual Basic

Add missing cases/default case/both

When creating a `switch` statement in C#, or `Select Case` statement in Visual Basic, you can use a Code Action to automatically add missing case items, a default case statement, or both.

Consider the following enumeration and empty `switch` or `Select Case` statement:

```
enum MyEnum  
{  
    Item1,  
    Item2,  
    Item3  
}  
  
...  
  
MyEnum myEnum = MyEnum.Item1;  
  
switch(myEnum)  
{  
}
```

```

Enum MyEnum
    Item1
    Item2
    Item3
End Enum

...

Dim myEnum as MyEnum = MyEnum.Item1

Select Case myEnum
End Select

```

Using the **Add Both** Quick Action fills in missing cases and adds a default case:

```

switch(myEnum)
{
    case MyEnum.Item1:
        break;
    case MyEnum.Item2:
        break;
    case MyEnum.Item3:
        break;
    default:
        break;
}

```

```

Select Case myEnum
    Case MyEnum.Item1
        Exit Select
    Case MyEnum.Item2
        Exit Select
    Case Else
        Exit Select
End Select

```

DIAGNOSTIC ID	APPLICABLE LANGUAGES	SUPPORTED VERSION
IDE0010	C# and Visual Basic	Visual Studio 2017 version 15.3 and later

Add null checks for parameters

This Quick Action enables you to add a check in your code to tell whether a parameter is null.

```

// Before
class MyClass
{
    public string MyProperty { get; set; }

    public MyClass(string myProperty) // cursor inside myProperty
    {
        MyProperty = myProperty;
    }
}

// Add null check

// After
class MyClass
{
    public string MyProperty { get; set; }

    public MyClass(string myProperty)
    {
        MyProperty = myProperty ?? throw new ArgumentNullException(nameof(myProperty));
    }
}

```

APPLICABLE LANGUAGES	SUPPORTED VERSION
C# and Visual Basic	Visual Studio 2017 version 15.3 and later

Add argument name

```

// Before
var date = new DateTime(1997, 7, 8);

// Include argument name 'year' (include trailing arguments)

// After
var date = new DateTime(year: 1997, month: 7, day: 8);

```

APPLICABLE LANGUAGES	SUPPORTED VERSION
C# and Visual Basic	Visual Studio 2017 version 15.3 and later

Add braces

The Add braces Quick Action wraps braces around single-line `if` statements.

```

// Before
if (true)
    return "hello,world";

// Add braces

// After
if (true)
{
    return "hello,world";
}

```

DIAGNOSTIC ID	APPLICABLE LANGUAGES	SUPPORTED VERSION
IDE0011	C#	Visual Studio 2017 and later

Add and order modifiers

These Quick Actions help organize modifiers by enabling you to sort existing and add missing accessibility modifiers.

```
// Before
enum Color
{
    Red, White, Blue
}

// Add accessibility modifiers

// After
internal enum Color
{
    Red, White, Blue
}
```

```
// Before
static private int thisFieldIsPublic;

// Order modifiers

// After
private static int thisFieldIsPublic;
```

DIAGNOSTIC ID	APPLICABLE LANGUAGES	SUPPORTED VERSION
IDE0036	C# and Visual Basic	Visual Studio 2017 version 15.5 and later
IDE0040	C# and Visual Basic	Visual Studio 2017 version 15.5 and later

Code transformations

Convert 'if' construct to 'switch'

This Quick Action enables you to convert an **if-then-else** construct to a **switch** construct.

```

// Before
if (obj is string s)
{
    Console.WriteLine("obj is a string: " + s);
}

else if (obj is int i && i > 10)
{
    Console.WriteLine("obj is an int greater than 10");
}

// Convert to switch

// After
switch (obj)
{
    case string s:
        Console.WriteLine("Obj is a string: " + s);
        break;
    case int i when i > 10:
        Console.WriteLine("obj is an int greater than 10");
        break;
}

```

```

' Before
If TypeOf obj Is String s Then
    Console.WriteLine("obj is a string: " + s)
Else If TypeOf obj Is Integer i And i > 10 Then
    Console.WriteLine("obj is an int greater than 10")
End If

' Convert to switch

' After
Select Case obj
    Case String s
        Console.WriteLine("Obj is a string: " + s)
        Exit Sub
    Case Integer i when i > 10
        Console.WriteLine("obj is an int greater than 10")
        Exit Sub
End Select

```

APPLICABLE LANGUAGES	SUPPORTED VERSION
C# and Visual Basic	Visual Studio 2017 version 15.3 and later

Convert to interpolated string

[Interpolated strings](#) are an easy way to express strings with embedded variables, similar to the [String.Format](#) method. This Quick Action recognizes cases where strings are concatenated, or using [String.Format](#), and changes the usage to an interpolated string.

```
// Before
int num = 3;
string s = string.Format("My string with {0} in the middle", num);

// Convert to interpolated string

// After
int num = 3;
string s = $"My string with {num} in the middle";
```

```
' Before
Dim num as Integer = 3
Dim s as String = String.Format("My string with {0} in the middle", num)

' Convert to interpolated string

' After
Dim num as Integer = 3
Dim s As String = $"My string with {num} in the middle"
```

APPLICABLE LANGUAGES	SUPPORTED VERSION
C# 6.0+ and Visual Basic 14+	Visual Studio 2017 and later

Use object initializers

This Quick Action enables you to use [object initializers](#) rather than invoking the constructor and having additional lines of assignment statements.

```
// Before
var c = new Customer();
c.Age = 21;

// Object initialization can be simplified

// After
var c = new Customer() { Age = 21 };
```

```
' Before
Dim c = New Customer()
c.Age = 21

' Object initialization can be simplified

' After
Dim c = New Customer() With {.Age = 21}
```

DIAGNOSTIC ID	APPLICABLE LANGUAGES	SUPPORTED VERSION
IDE0017	C# and Visual Basic	Visual Studio 2017 and later

Use collection initializers

This Quick Action lets you use [collection initializers](#) rather than multiple calls to the `Add` method of your class.

```
// Before
var list = new List<int>();
list.Add(1);
list.Add(2);
list.Add(3);

// Collection initialization can be simplified

// After
var list = new List<int> { 1, 2, 3};
```

```
' Before
Dim list = New List(Of Integer)
list.Add(1)
list.Add(2)
list.Add(3)

' Collection initialization can be simplified

' After
Dim list = New List(Of Integer) From {1, 2, 3}
```

DIAGNOSTIC ID	APPLICABLE LANGUAGES	SUPPORTED VERSION
IDE0028	C# and Visual Basic	Visual Studio 2017 and later

Convert auto property to full property

This Quick Action enables you to convert an auto property to a full property, and vice versa.

```
// Before
private int MyProperty { get; set; }

// Convert to full property

// After
private int MyProperty
{
    get { return _myProperty; }
    set { _myProperty = value; }
}
```

```
' Before
Public Property Name As String

' Convert to full property

' After
Private _Name As String

Public Property Name As String
    Get
        Return _Name
    End Get
    Set
        _Name = Value
    End Set
End Property
```

APPLICABLE LANGUAGES	SUPPORTED VERSION
C# and Visual Basic	Visual Studio 2017 version 15.5 and later

Convert block body to expression-bodied member

This Quick Action allows you to convert block bodies into expression-bodied members for methods, constructors, operators, properties, indexers, and accessors.

```
//Before
class MyClass4
{
    private int _myProperty;

    public int MyProperty
    {
        get { return _myProperty; }
        set
        {
            _myProperty = value;
        }
    }

    public MyClass4(int myProperty)
    {
        MyProperty = myProperty;
    }

    public void PrintProperty()
    {
        Console.WriteLine(MyProperty);
    }
}

// Use expression body for accessors/constructors/methods

// After
class MyClass4
{
    private int _myProperty;

    public int MyProperty
    {
        get => _myProperty;
        set => _myProperty = value;
    }

    public MyClass4(int myProperty) => MyProperty = myProperty;

    public void PrintProperty() => Console.WriteLine(MyProperty);
}
```

DIAGNOSTIC ID	APPLICABLE LANGUAGES	SUPPORTED VERSION
IDE0021-27	C# 6.0+	Visual Studio 2017 and later

Convert anonymous function to local function

This Quick Action converts anonymous functions into local functions.

```

// Before
Func<int, int> fibonacci = null;
fibonacci = (int n) =>
{
    return n <= 1 ? 1 : fibonacci(n - 1) + fibonacci(n - 2);
};

// Use local function

// After
int fibonacci(int n)
{
    return n <= 1 ? 1 : fibonacci(n-1) + fibonacci(n-2);
}

```

Convert 'ReferenceEquals' to 'is null'

DIAGNOSTIC ID	APPLICABLE LANGUAGES	SUPPORTED VERSION
IDE0041	C# 7.0+	Visual Studio 2017 version 15.5 and later

This Quick Action suggests the use of [pattern matching](#) rather than the `ReferenceEquals` coding-pattern, where possible.

```

// Before
var value = "someString";
if (object.ReferenceEquals(value, null))
{
    return;
}

// Use 'is null' check

// After
var value = "someString";
if (value is null)
{
    return;
}

```

DIAGNOSTIC ID	APPLICABLE LANGUAGES	SUPPORTED VERSION
IDE0039	C# 7.0+	Visual Studio 2017 version 15. and later

Introduce pattern matching

This Quick Action suggests the use of [pattern matching](#) with casts and null checks in C#.

```
// Before
if (o is int)
{
    var i = (int)o;
    ...
}

// Use pattern matching
```

```
// After
if (o is int i)
{
    ...
}
```

```
// Before
var s = o as string;
if (s != null)
{
    ...
}

// Use pattern matching
```

```
// After
if (o is string s)
{
    ...
}
```

DIAGNOSTIC ID	APPLICABLE LANGUAGES	SUPPORTED VERSION
IDE0020	C# 7.0+	Visual Studio 2017 and later
IDE0019	C# 7.0+	Visual Studio 2017 and later

Change base for numeric literals

This Quick Action enables you to convert a numeric literal from one base numeric system to another. For example, you can change a number to hexadecimal or to binary format.

```
// Before
int countdown = 2097152;

// Convert to hex

// After
int countdown = 0x200000;
```

```
' Before
Dim countdown As Integer = 2097152

' Convert to hex

' After
Dim countdown As Integer = &H200000
```

APPLICABLE LANGUAGES	SUPPORTED VERSION
C# 7.0+ and Visual Basic 14+	Visual Studio 2017 version 15.3 and later

Insert digit separators into literals

This Quick Action enables you to add separator characters into literal values.

```
// Before
int countdown = 1000000;

// Separate thousands

// After
int countdown = 1_000_000;
```

```
' Before
Dim countdown As Integer = 1000000

' Separate thousands

' After
Dim countdown As Integer = 1_000_000
```

APPLICABLE LANGUAGES	SUPPORTED VERSION
C# 7.0+ and Visual Basic 14+	Visual Studio 2017 version 15.3 and later

Use explicit tuple names

This Quick Action identifies areas where the explicit tuple name can be used rather than Item1, Item2, etc.

```
// Before
(string name, int age) customer = GetCustomer();
var name = customer.Item1;

// Use explicit tuple name

// After
(string name, int age) customer = GetCustomer();
var name = customer.name;
```

```
' Before
Dim customer As (name As String, age As Integer) = GetCustomer()
Dim name = customer.Item1

' Use explicit tuple name

' After
Dim customer As (name As String, age As Integer) = GetCustomer()
Dim name = customer.name
```

DIAGNOSTIC ID	APPLICABLE LANGUAGES	SUPPORTED VERSION
IDE0033	C# 7.0+ and Visual Basic 15+	Visual Studio 2017 and later

Use inferred names

This Quick Action points out when code can be simplified to use inferred member names in anonymous types, or inferred element names in tuples.

```
// Before
var anon = new { age = age, name = name };

// Use inferred member name

// After
var anon = new { age, name };
```

```
// Before
var tuple = (age: age, name: name);

// Use inferred tuple element name

// After
var tuple = (age, name);
```

DIAGNOSTIC ID	APPLICABLE LANGUAGES	SUPPORTED VERSION
IDE0037	C#	Visual Studio 2017 version 15.5 and later
IDE0037	C# 7.1+	Visual Studio 2017 version 15.5 and later

Deconstruct tuple declaration

This Quick Action enables deconstructing tuple variable declarations.

```
// Before
var person = GetPersonTuple();
Console.WriteLine($"{person.name} {person.age}");

(int x, int y) point = GetPointTuple();
Console.WriteLine($"{point.x} {point.y}");

// Deconstruct variable declaration

// After
var (name, age) = GetPersonTuple();
Console.WriteLine($"{name} {age}");

(int x, int y) = GetPointTuple();
Console.WriteLine($"{x} {y}");
```

DIAGNOSTIC ID	APPLICABLE LANGUAGES	SUPPORTED VERSION
IDE0042	C# 7.0+	Visual Studio 2017 version 15.5 and later

Make method synchronous

When using the `async` or `Async` keyword on a method, it's expected that inside that method the `await` or `Await` keyword is also used. However, if this isn't the case, a Quick Action appears that makes the method synchronous by removing the `async` or `Async` keyword and changing the return type. Use the **Make method synchronous** option from the Quick Actions menu.

```
// Before
async Task<int> MyAsyncMethod()
{
    return 3;
}

// Make method synchronous

// After
int MyAsyncMethod()
{
    return 3;
}
```

```
' Before
Async Function MyAsyncMethod() As Task(Of Integer)
    Return 3
End Function

' Make method synchronous

' After
Function MyAsyncMethod() As Integer
    Return 3
End Function
```

ERROR ID	APPLICABLE LANGUAGES
CS1998, BC42356	C# and Visual Basic

Make method asynchronous

When using the `await` or `Await` keyword inside of a method, it's expected that the method is marked with the `async` or `Async` keyword. However, if this isn't the case, a Quick Action appears that makes the method asynchronous. Use the **Make method/Function asynchronous** option from the Quick Actions menu.

```
// Before
int MyAsyncMethod()
{
    return await Task.Run(...);
}

// Make method asynchronous

// After
async Task<int> MyAsyncMethod()
{
    return await Task.Run(...);
}
```

```
' Before
Function MyAsyncMethod() as Integer
    Return Await Task.Run(...)
End Function

' Make method asynchronous

' After
Async Function MyAsyncMethod() As Task(Of Integer)
    Return Await Task.Run(...)
End Function
```

ERROR ID	APPLICABLE LANGUAGES	SUPPORTED VERSION
CS4032, BC37057	C# and Visual Basic	Visual Studio 2017 and later

See also

- [Quick Actions](#)

Generate a class or type in Visual Studio

10/18/2019 • 2 minutes to read • [Edit Online](#)

This code generation applies to:

- C#
- Visual Basic

What: Lets you immediately generate the code for a class or type.

When: You introduce a new class or type and want to properly declare it, automatically.

Why: You could declare the class or type before using it, however this feature will generate the class or type automatically.

How-to

1. Place your cursor on the line where there is a red squiggle. The red squiggle indicates a class that doesn't yet exist.

- C#:

```
static void Main(string[] args)
{
    MyNewType t = new MyNewType();
}
```

- Visual Basic:

```
Sub Main()
    Dim t As New MyNewType.
End Sub
```

2. Next, do one of the following:

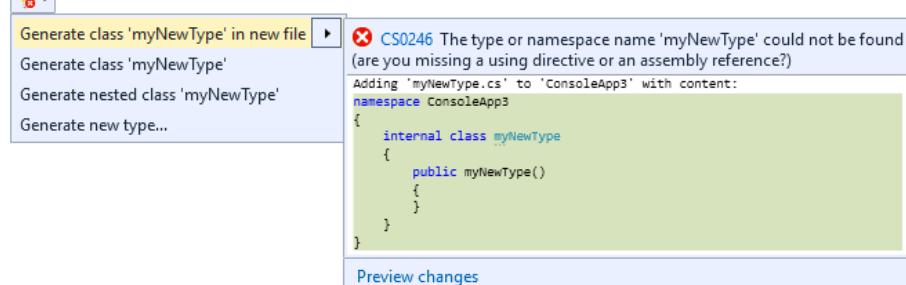
- **Keyboard**

- Press **Ctrl+.** to trigger the **Quick Actions and Refactorings** menu.

- **Mouse**

- Right-click and select the **Quick Actions and Refactorings** menu.
 - Hover over the red squiggle and click the  icon that appears.
 - Click the  icon that appears in the left margin if the text cursor is already on the line with the red squiggle.

```
static void Main(string[] args)
{
    myNewType t = new myNewType();
}
```



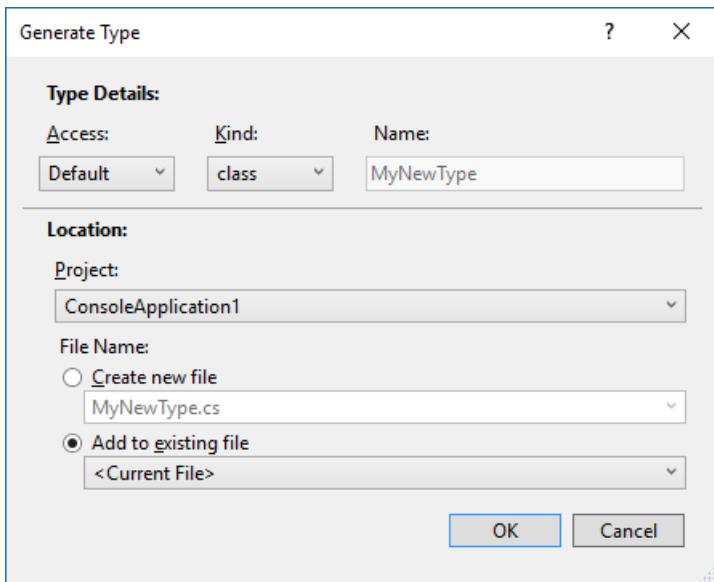
3. Select one of the options from the drop-down menu:

- Generate class '*TypeName*' in new file—Creates a class named *TypeName* in a file named *TypeName.cs/.vb*
- Generate class '*TypeName*'—Creates a class named *TypeName* in the current file.
- Generate nested class '*TypeName*'—Creates a class named *TypeName* nested inside the current class.
- Generate new type...—Creates a new class or struct with all of the properties you specify.

TIP

Use the **Preview changes** link at the bottom of the preview window to see all of the changes that will be made before making your selection.

4. If you selected the **Generate new type** item, the **Generate Type** dialog box opens. Configure the accessibility, kind, and location of the new type.



SELECTION	DESCRIPTION
Access	Set the type to have <i>Default</i> , <i>Internal</i> or <i>Public</i> access.
Kind	This can be set as <i>class</i> or <i>struct</i> .
Name	This cannot be changed and will be the name you already typed.
Project	If there are multiple projects in your solution, you can choose where you want the class/struct to live.
File Name	You can create a new file or you can add the type to an existing file.

The class or struct is created. For C#, a constructor is also created.

- C#

```
class MyNewType
{
    public MyNewType()
    {
    }
}
```

- Visual Basic

```
Friend Class MyNewType
End Class
```

See also

- [Code Generation](#)
- [Preview Changes](#)

Generate a method in Visual Studio

10/18/2019 • 2 minutes to read • [Edit Online](#)

This code generation applies to:

- C#
- Visual Basic

What: Lets you immediately add a method to a class.

When: You introduce a new method and want to properly declare it, automatically.

Why: You could declare the method and parameters before using it, however this feature will generate the declaration automatically.

How-to

1. Place your cursor on the line where there is a red squiggle. The red squiggle indicates a method that doesn't exist yet.

- C#:

```
static void Main(string[] args)
{
    MyNewType t = new MyNewType();
    t.MyMethod(12345);
}
```

- Visual Basic:

```
Sub Main()
    Dim t As New MyNewType
    t.MyMethod(12345)
End Sub
```

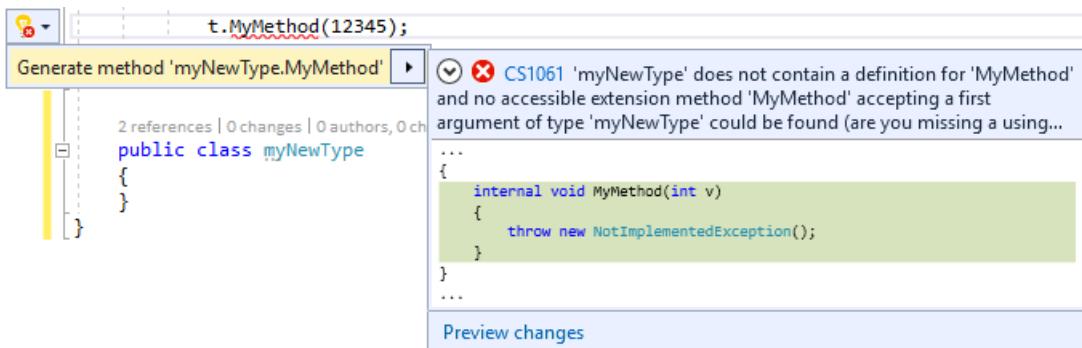
2. Next, do one of the following:

- **Keyboard**

- Press **Ctrl+.** to trigger the **Quick Actions and Refactorings** menu.

- **Mouse**

- Right-click and select the **Quick Actions and Refactorings** menu.
 - Hover over the red squiggle and click the  icon that appears.
 - Click the  icon that appears in the left margin if the text cursor is already on the line with the red squiggle.



3. Select **Generate method** from the drop-down menu.

TIP

Use the **Preview changes** link at the bottom of the preview window to see all of the changes that will be made before making your selection.

The method is created with any parameters inferred from its usage.

- C#:

```
internal void MyMethod(Int32 v)
{
    throw new NotImplementedException();
}
```

- Visual Basic:

```
Friend Sub MyMethod(v As Integer)
    Throw New NotImplementedException()
End Sub
```

See also

- [Code Generation](#)
- [Preview Changes](#)

Generate a field, property, or local variable in Visual Studio

10/18/2019 • 2 minutes to read • [Edit Online](#)

This code generation applies to:

- C#
- Visual Basic

What: Lets you immediately generate the code for a previously undeclared field, property, or local.

When: You introduce a new field, property or local while typing and want to properly declare it, automatically.

Why: You could declare the field, property or local before using it, however this feature will generate the declaration and type automatically.

How-to

1. Place your cursor on the line where there is a red squiggle. The red squiggle indicates a field, local or property that doesn't yet exist.

- C#:

```
static void Main(string[] args)
{
    double area = Math.PI * radius * radius;
```

- Visual Basic:

```
Sub Main()
    Dim area As Double = Math.PI * radius * radius
End Sub
```

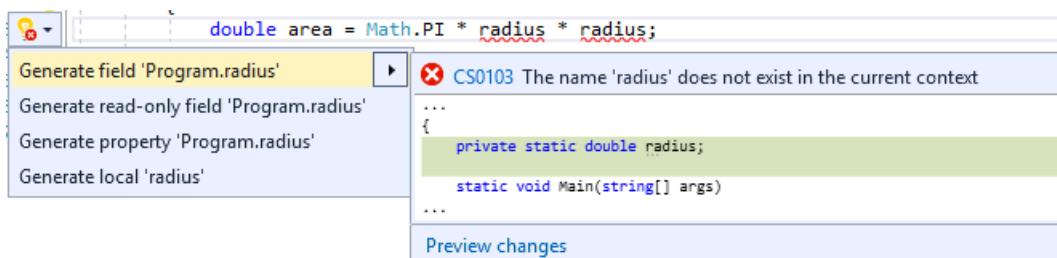
2. Next, do one of the following:

- **Keyboard**

- Press **Ctrl+.** to trigger the **Quick Actions and Refactorings** menu.

- **Mouse**

- Right-click and select the **Quick Actions and Refactorings** menu.
 - Hover over the red squiggle and click the  icon that appears.
 - Click the  icon that appears in the left margin if the text cursor is already on the line with the red squiggle.



3. Select one of the generation options from the drop-down menu.

TIP

Use the **Preview changes** link at the bottom of the preview window to see all of the changes that will be made before making your selection.

The field, property or local is created, with the type inferred from its usage.

- C#:

```
static void Main(string[] args)
{
    Int32 radius = 0;
    double area = Math.PI * radius * radius;
}
```

- Visual Basic:

```
Sub Main()
    Dim radius As Double = Nothing
    Dim area As Double = Math.PI * radius * radius
End Sub
```

See also

- [Code Generation](#)
- [Preview Changes](#)

Generate a constructor in Visual Studio

10/18/2019 • 3 minutes to read • [Edit Online](#)

This code generation applies to:

- C#
- Visual Basic

What: Lets you immediately generate the code for a new constructor on a class.

When: You introduce a new constructor and want to properly declare it automatically, or you modify an existing constructor.

Why: You could declare the constructor before using it, however this feature will generate it, with the proper parameters, automatically. Furthermore, modifying an existing constructor requires updating all the callsites unless you use this feature to update them automatically.

How: There are several ways to generate a constructor:

- [Generate constructor and pick members](#)
- [Generate constructor from selected fields](#)
- [Generate constructor from new usage](#)
- [Add parameter to existing constructor](#)
- [Create and initialize field/property from a constructor parameter](#)

Generate constructor and pick members (C# only)

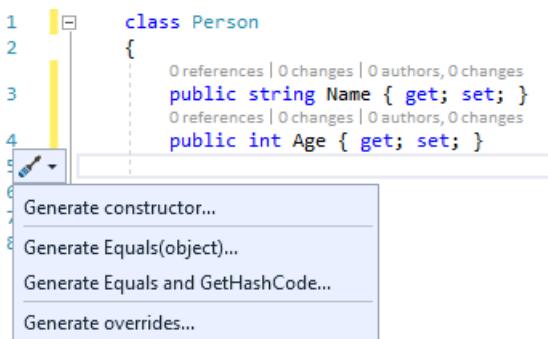
1. Place your cursor in any empty line in a class:

```
class Person
{
    public string Name { get; set; }
    public int Age { get; set; }

}
```

2. Next, do one of the following:

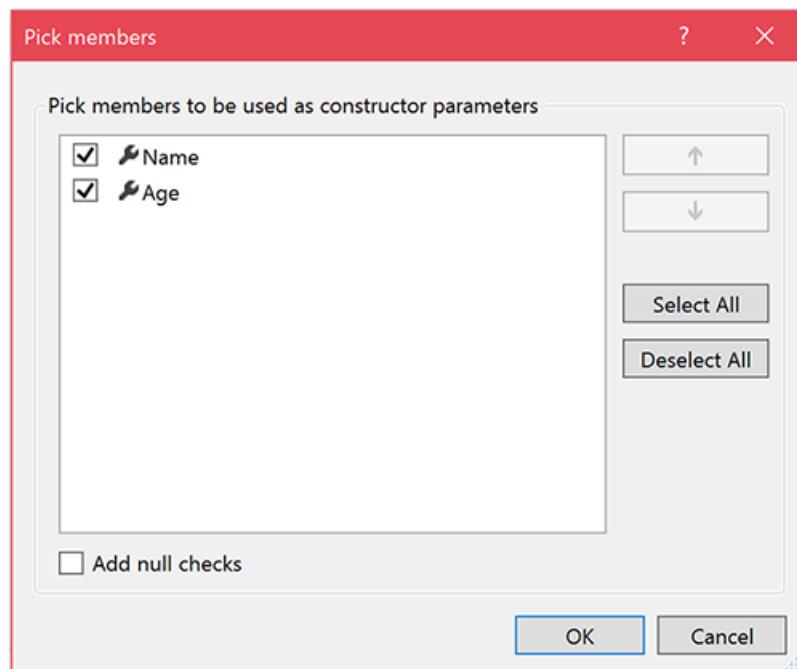
- **Keyboard**
 - Press **Ctrl+.** to trigger the **Quick Actions and Refactorings** menu.
- **Mouse**
 - Right-click and select the **Quick Actions and Refactorings** menu.
 - Click the  icon that appears in the left margin if the text cursor is already on the empty line in the class.



3. Select **Generate constructor** from the drop-down menu.

The **Pick members** dialog box opens.

4. Pick the members you want to include as constructor parameters. You can order them using the up and down arrows. Choose **OK**.



TIP

You can check the **Add null checks** checkbox to automatically generate null checks for your constructor parameters.

The constructor is created with the specified parameters.

```
class Person
{
    public string Name { get; set; }
    public int Age { get; set; }

    public Person(string name, int age)
    {
        Name=name;
        Age=age;
    }
}
```

Generate constructor from selected fields (C# only)

1. Highlight the members you wish to have in your generated constructor:

```
class Person
{
    public string Name { get; set; }
    public string Surname { get; set; }
    public int Age { get; set; }
}
```

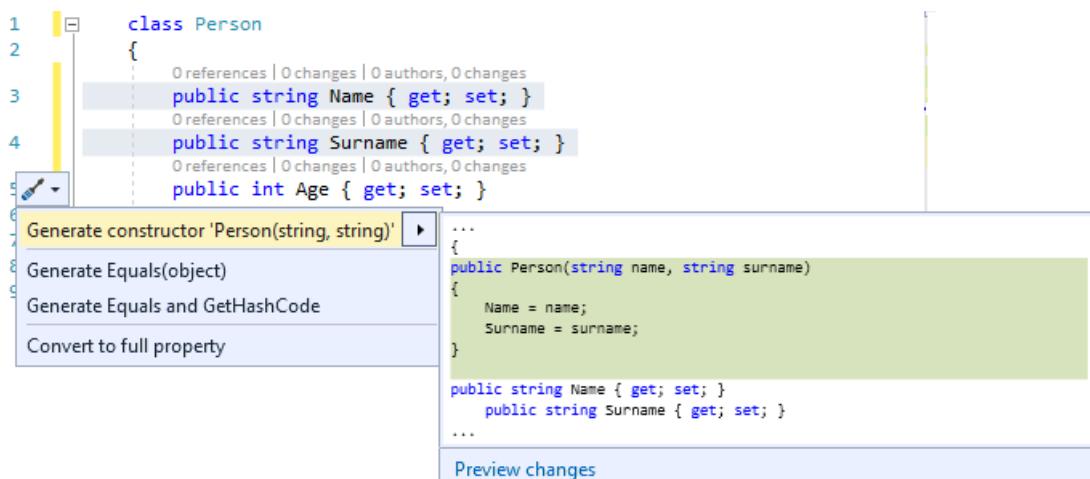
2. Next, do one of the following:

- **Keyboard**

- Press **Ctrl+.** to trigger the **Quick Actions and Refactorings** menu.

- **Mouse**

- Right-click and select the **Quick Actions and Refactorings** menu.
 - Click the  icon that appears in the left margin if the text cursor is already on the line with the selection.



3. Select **Generate constructor 'TypeName(..)'** from the drop-down menu.

The constructor is created with the selected parameters.

```
class Person
{
    public Person(string name, string surname)
    {
        Name=name;
        Surname=surname;
    }

    public string Name { get; set; }
    public string Surname { get; set; }
    public int Age { get; set; }
}
```

Generate constructor from new usage (C# and Visual Basic)

1. Place your cursor on the line where there is a red squiggle. The red squiggle indicates a call to a constructor that doesn't yet exist.

- C#:

```
static void Main(string[] args)
{
    Person p = new Person("Bob", "Jones");
}
```

- Visual Basic:

```
Sub Main()
    Dim p As New Person("Bob", "Jones")
End Sub
```

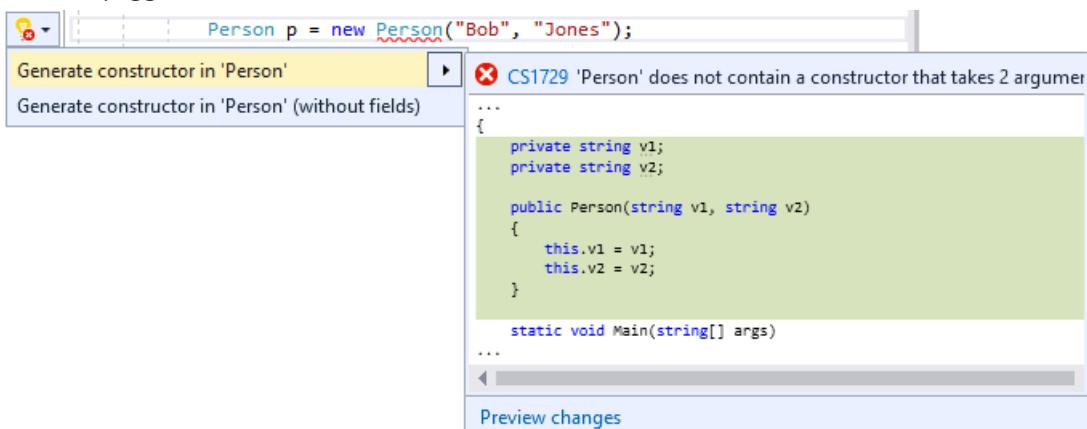
2. Next, do one of the following:

- **Keyboard**

- Press **Ctrl+.** to trigger the **Quick Actions and Refactorings** menu.

- **Mouse**

- Right-click and select the **Quick Actions and Refactorings** menu.
- Hover over the red squiggle and click the icon that appears.
- Click the icon that appears in the left margin if the text cursor is already on the line with the red squiggle.



3. Select **Generate constructor in 'TypeName'** from the drop-down menu.

TIP

Use the **Preview changes** link at the bottom of the preview window to see all of the changes that will be made before making your selection.

The constructor is created, with any parameters inferred from its usage.

- C#:

```
class Person
{
    private String v1;
    private String v2;

    public Person(String v1, String v2)
    {
        this.v1 = v1;
        this.v2 = v2;
    }
}
```

- Visual Basic:

```

Class Person
    Private v1 As String
    Private v2 As String

    Public Sub New(v1 As String, v2 As String)
        Me.v1 = v1
        Me.v2 = v2
    End Sub

    Public Property FirstName As String
    Public Property LastName As String
End Class

```

Add parameter to existing constructor (C# only)

1. Add a parameter to an existing constructor call.
2. Place your cursor on the line where there is a red squiggle indicating you've used a constructor that doesn't yet exist.

```

class Program
{
    static void Main(string[] args)
    {
        var p = new Person("John", "Smith", 30);
    }
}

class Person
{
    public Person(string name, string surname)
    {
        Name=name;
        Surname=surname;
    }

    public string Name { get; set; }
    public string Surname { get; set; }
}

```

3. Next, do one of the following:

- **Keyboard**
 - Press **Ctrl+.** to trigger the **Quick Actions and Refactorings** menu.
- **Mouse**
 - Right-click and select the **Quick Actions and Refactorings** menu.
 - Hover over the red squiggle and click the  icon that appears.
 - Click the  icon that appears in the left margin if the text cursor is already on the line with the red squiggle.

The screenshot shows a code editor with C# code. A tooltip is open over the line `var p = new Person("John", "Smith", 30);` with the text "Add parameter to 'Person(string, string)'". Below the tooltip, a red error bar highlights the line `var p = new Person("John", "Smith", 30);` with the message "CS1729 'Person' does not contain a constructor that takes 3 arguments". The code editor shows the following code:

```

0 references | 0 changes | 0 authors, 0 changes
class Program
{
    0 references | 0 changes | 0 authors, 0 changes
    static void Main(string[] args)
    {
        var p = new Person("John", "Smith", 30);
    }
}
2 references | 0 changes | 0 authors, 0 changes
class Person
{
    1 reference | 0 changes
    public Person(string name, string surname)
    {
        Name = name;
        Surname = surname;
    }
    1 reference | 0 changes | 0 authors, 0 changes
    public string Name { get; set; }
    1 reference | 0 changes | 0 authors, 0 changes
    public string Surname { get; set; }
}

```

4. Select **Add parameter to 'TypeName(...)'** from the drop-down menu.

The parameter is added to the constructor, with its type inferred from its usage.

```

class Program
{
    static void Main(string[] args)
    {
        var p = new Person("John", "Smith", 30);
    }
}

class Person
{
    public Person(string name, string surname, int v)
    {
        Name=name;
        Surname=surname;
    }

    public string Name { get; set; }
    public string Surname { get; set; }
}

```

You can also add a parameter to an existing method. For more information, see [Add parameter to a method](#).

Create and initialize a field or property from a constructor parameter (C# only)

1. Find an existing constructor, and add a parameter:

```

class Person
{
    public Person(string name, string surname, int age)
    {
        Name=name;
        Surname=surname;
    }

    public string Name { get; set; }
    public string Surname { get; set; }
}

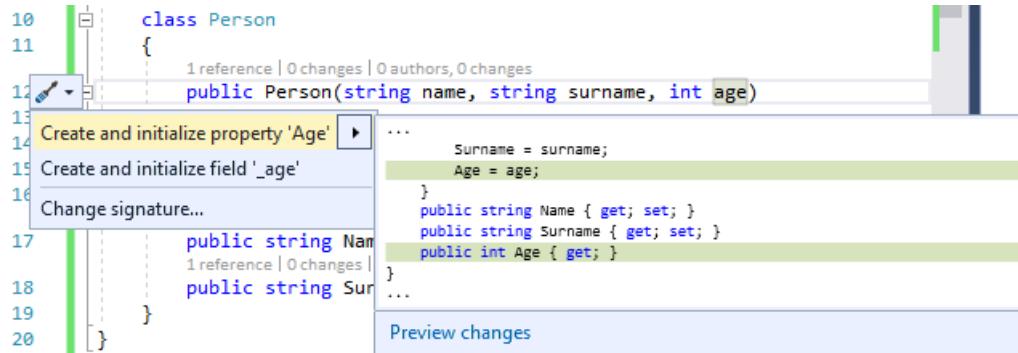
```

2. Place your cursor inside the newly added parameter.

3. Next, do one of the following:

- **Keyboard**

- Press **Ctrl+.** to trigger the **Quick Actions and Refactorings** menu.
- **Mouse**
 - Right-click and select the **Quick Actions and Refactorings** menu.
 - Click the  icon that appears in the left margin if the text cursor is already on the line with the added parameter.



4. Select **Create and initialize property** or **Create and initialize field** from the drop-down menu.

The field or property is declared and automatically named to match your types. A line of code is also added to initialize the field or property in the constructor body.

```

class Person
{
    public Person(string name, string surname, int age)
    {
        Name=name;
        Surname=surname;
        Age=age;
    }

    public string Name { get; set; }
    public string Surname { get; set; }
    public int Age { get; }
}

```

See also

- [Code generation](#)
- [Preview changes](#)

Generate a deconstructor in Visual Studio

5/10/2019 • 2 minutes to read • [Edit Online](#)

This code generation applies to:

- C#

What: Lets you immediately generate the method stub for a new deconstructor.

When: You want to properly deconstruct your type automatically.

Why: You can manually type a deconstructor, but this feature generates the stub for you with the correct out parameters.

Generate a deconstructor

1. Declare a new type with the desired out parameters specified. This declaration will cause an error when no deconstruct instance matching your declaration can be found.

```
class Class
{
    0 references
    public void GenerateDeconstructMethod()
    {
        (int x, int y) = new MyInternalClass();
    }
    2 references
    private class MyInternalClass
    {
        1 reference
        public MyInternalClass()
        {
        }
    }
}
```

2. Take one of the following steps:

- **Keyboard**

- With the cursor in your declaration, select Ctrl+. to trigger the **Quick Actions and Refactorings** menu.

- **Mouse**

- Right-click and select the **Quick Actions and Refactorings** menu.
 - Select the  icon that appears in the left margin if the text cursor is already on the empty line in the class.



3. Select **Generate method 'MyInternalClass.Deconstruct'** to generate the deconstructor.

```
class Class
{
    0 references
    public void GenerateDeconstructMethod()
    {
        (int x, int y) = new MyInternalClass();
    }
    2 references
    private class MyInternalClass
    {
        1 reference
        public MyInternalClass()
        {
        }

        1 reference
        internal void Deconstruct(out int x, out int y)
        {
            throw new NotImplementedException();
        }
    }
}
```

See also

- [Code generation](#)
- [Preview changes](#)
- [Tips for .NET developers](#)

Add a parameter to a method using a Quick Action

10/18/2019 • 2 minutes to read • [Edit Online](#)

This code generation applies to:

- C#
- Visual Basic

What: Lets you automatically add a parameter to a method, based on usage.

When: You need to add a parameter to a method and want to properly declare it automatically.

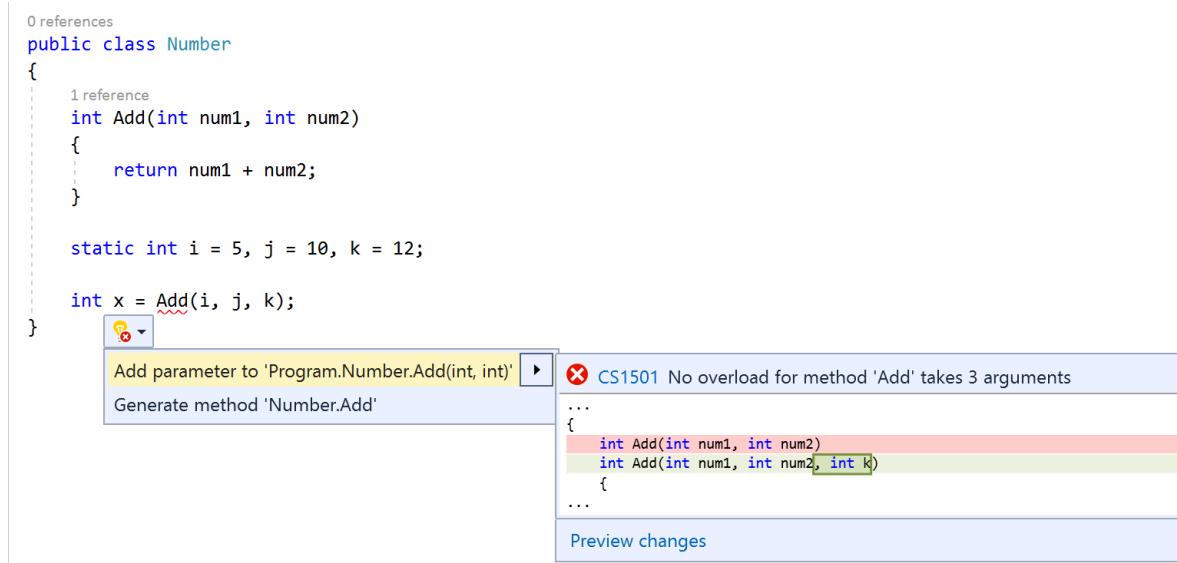
Why: You could add the parameter to the method declaration before calling it, however this feature adds it automatically based on a method call.

How to use it

1. Add an extra argument to a method call.

A red squiggle appears under the name of the method where you call it.

2. Place your pointer over the red squiggle until the Quick Actions menu appears. Select the **down arrow** on the Quick Actions menu, and then select **Add parameter to [method]**.



TIP

You can also access the Quick Actions menu by placing your cursor on the line of the method call, and then either pressing **Ctrl+.** (period) or selecting the light bulb icon in the file margin.

Visual Studio adds the new parameter to the method declaration.

NOTE

If you have other calls to the method, they may produce errors after you use this Quick Action, because they don't specify an argument for the newly added parameter.

See also

- [Add parameter to constructor](#)

Generate an override in Visual Studio

10/18/2019 • 2 minutes to read • [Edit Online](#)

This code generation applies to:

- C#
- Visual Basic

What: Lets you immediately generate the code for any method which can be overridden from a base class.

When: You want to override a base class method and generate the signature automatically.

Why: You could write the method signature yourself, however this feature will generate the signature automatically.

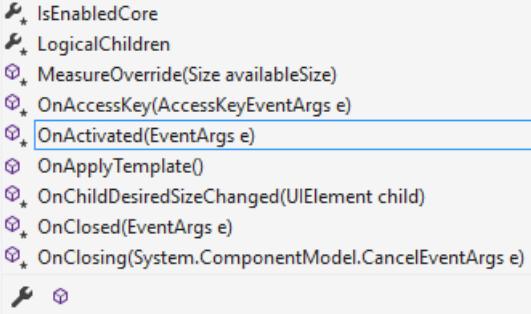
How-to

1. Type `override` in C# or `Overrides` in Visual Basic, followed by a space, where you would like to insert an override method.

- C#:

```
public partial class MainWindow : Window
{
    public MainWindow()
    {
        InitializeComponent();
    }

    override
}
```



The screenshot shows the Visual Studio code editor with the following code:

```
public partial class MainWindow : Window
{
    public MainWindow()
    {
        InitializeComponent();
    }

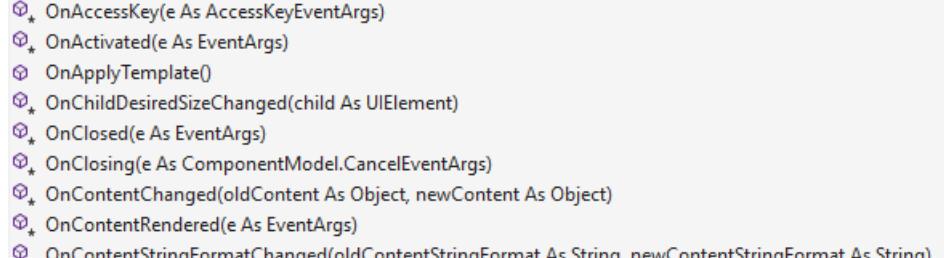
    override
}
```

A completion list is displayed below the cursor, showing various methods from the `Window` base class. The method `OnActivated(EventArgs e)` is highlighted with a blue selection bar.

- Visual Basic:

```
Class MainWindow

    Overrides ~
End Class
```



The screenshot shows the Visual Studio code editor with the following code:

```
Class MainWindow

    Overrides ~
End Class
```

A completion list is displayed below the cursor, showing various methods from the `Window` base class. The method `OnActivated(e As EventArgs)` is highlighted with a blue selection bar.

2. Select the method you want to override from the base class.

TIP

- Use the property icon  to show or hide properties in the list.
- Use the method icon  to show or hide methods in the list.

The selected method or property is added to the class as an override, ready to be implemented.

- C#:

```
public partial class MainWindow : Window
{
    public MainWindow()
    {
        InitializeComponent();
    }

    protected override void OnActivated(EventArgs e)
    {
        base.OnActivated(e);
    }
}
```

- Visual Basic:

```
Class MainWindow

    Protected Overrides Sub OnActivated(e As EventArgs)
        MyBase.OnActivated(e)
    End Sub

End Class
```

See also

- [Code Generation](#)

Generate Equals and GetHashCode method overrides in Visual Studio

10/18/2019 • 2 minutes to read • [Edit Online](#)

This code generation applies to:

- C#

What: Lets you generate **Equals** and **GetHashCode** methods.

When: Generate these overrides when you have a type that should be compared by one or more fields, instead of by object location in memory.

Why:

- If you are implementing a value type, you should consider overriding the **Equals** method to gain increased performance over the default implementation of the Equals method on ValueType.
- If you are implementing a reference type, you should consider overriding the **Equals** method if your type looks like a base type, such as Point, String, BigNumber, and so on.
- Override the **GetHashCode** method to allow a type to work correctly in a hash table. Read more guidance on [equality operators](#).

How-to

1. Place your cursor somewhere on the line of your type declaration.

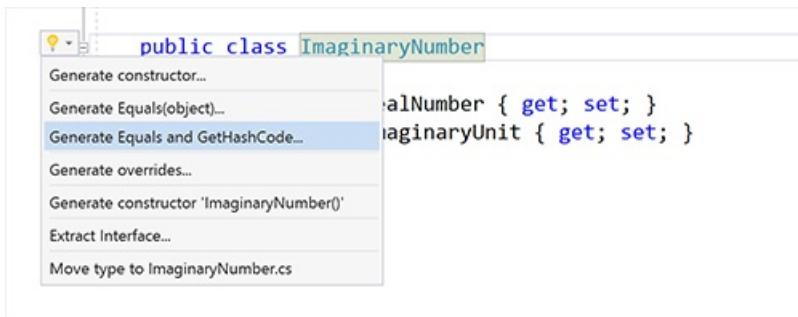
```
public class ImaginaryNumber
{
    public double RealNumber { get; set; }
    public double ImaginaryUnit { get; set; }
}
```

TIP

Do not double-click select the type name, or the menu option won't be available. Just place the cursor somewhere on the line.

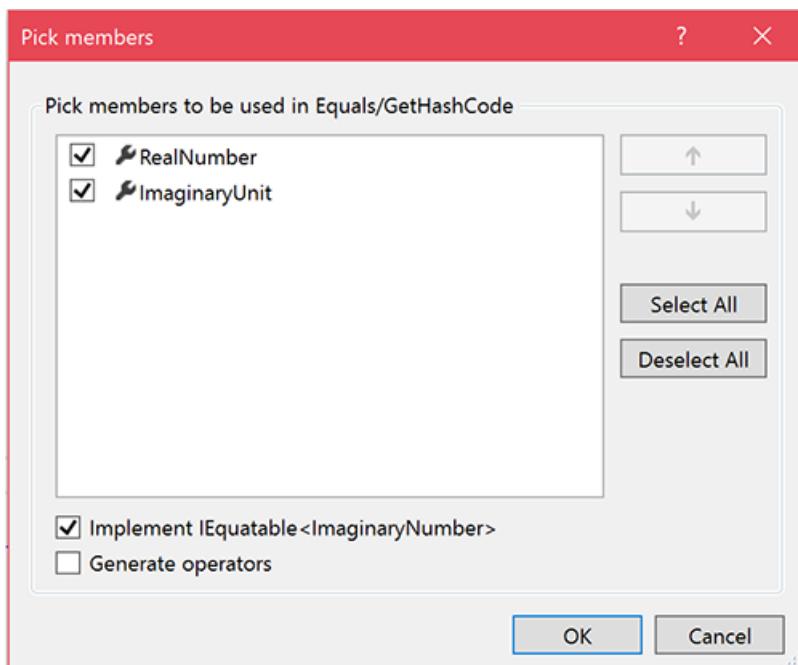
2. Next, do one of the following:

- Press **Ctrl + .** to trigger the **Quick Actions and Refactorings** menu.
- Right-click and select the **Quick Actions and Refactorings** menu.
- Click the  icon that appears in the left margin.



3. Select **Generate Equals(object)** or **Generate Equals and GetHashCode** from the drop-down menu.

4. In the **Pick members** dialog box, select the members you want to generate the methods for:



TIP

You can also choose to generate operators from this dialog by using the checkbox near the bottom of the dialog.

The `Equals` and `GetHashCode` methods are generated with default implementations.

```
public class ImaginaryNumber : IEquatable<ImaginaryNumber>
{
    public double RealNumber { get; set; }
    public double ImaginaryUnit { get; set; }

    public override bool Equals(object obj)
    {
        return Equals(obj as ImaginaryNumber);
    }

    public bool Equals(ImaginaryNumber other)
    {
        return other != null && RealNumber == other.RealNumber
            && ImaginaryUnit == other.ImaginaryUnit;
    }

    public override int GetHashCode()
    {
        var hashCode = 352033288;
        hashCode = hashCode * -1521134295 + RealNumber.GetHashCode();
        hashCode = hashCode * -1521134295 + ImaginaryUnit.GetHashCode();
        return hashCode;
    }
}
```

See also

- [Code Generation](#)
- [Preview Changes](#)

Add missing usings in Visual Studio

10/18/2019 • 2 minutes to read • [Edit Online](#)

This code generation applies to:

- C#

What: Lets you immediately add the necessary imports or [using directives](#) for copy-and-pasted code.

When: It's common practice to copy code from different places in your project or other sources and paste it in to new code. This Quick Action finds missing imports directives for copy-and-pasted code and then prompts you to add them.

Why: Because the Quick Action automatically adds necessary imports, you don't need to manually copy the `using` directives that your code needs.

Add missing usings refactoring

1. Copy code from a file and paste it into a new one without including the necessary `using` directives. The resulting error is accompanied by a code fix that adds the missing `using` directives.

NOTE

You need to enable this suggestion in **Tools > Options > Text Editor > C# > Advanced > Using Directives**.

2. Select **Ctrl+.** to open the **Quick Actions and Refactorings** menu.

The screenshot shows a code editor with the following snippet:

```
using System;
0 references
class Class
{
    0 references
    public void MyMethod()
    {
        string json = JsonConvert.SerializeObject("some json");
    }
}
```

A code error is shown at the line `string json = JsonConvert.SerializeObject("some json");`: **CS0103** The name 'JsonConvert' does not exist in the current context. A tooltip provides the error message and shows a code fix:

CS0103 The name 'JsonConvert' does not exist in the current context
using Newtonsoft.Json;
Newtonsoft.Json.JsonConvert
Generate variable 'JsonConvert' ▾
Generate type 'JsonConvert' ▾

The tooltip also includes a preview of the changes:

Preview changes

3. Select **using <your reference>**; to add the missing reference.

```
using Newtonsoft.Json;
using System;

0 references
class Class
{
    0 references
    public void MyMethod()
    {
        string json = JsonConvert.SerializeObject("some json");
    }
}
```

See also

- [Code generation](#)
- [Preview Changes](#)
- [Tips for .NET developers](#)

Implement an abstract class in Visual Studio

10/18/2019 • 2 minutes to read • [Edit Online](#)

This code generation applies to:

- C#
- Visual Basic

What: Lets you immediately generate the code required to implement an abstract class.

When: You want to inherit from an abstract class.

Why: You could manually implement all abstract members one-by-one, however this feature will generate all method signatures automatically.

How-to

1. Place your cursor on the line where there is a red squiggle that indicates you have inherited from an abstract class, but have not implemented all required members.

- C#:

```
abstract class MyAbstractClass
{
    public abstract void Method1();
    public abstract int Method2(int value);
}

class MyClass : MyAbstractClass
{
}
```

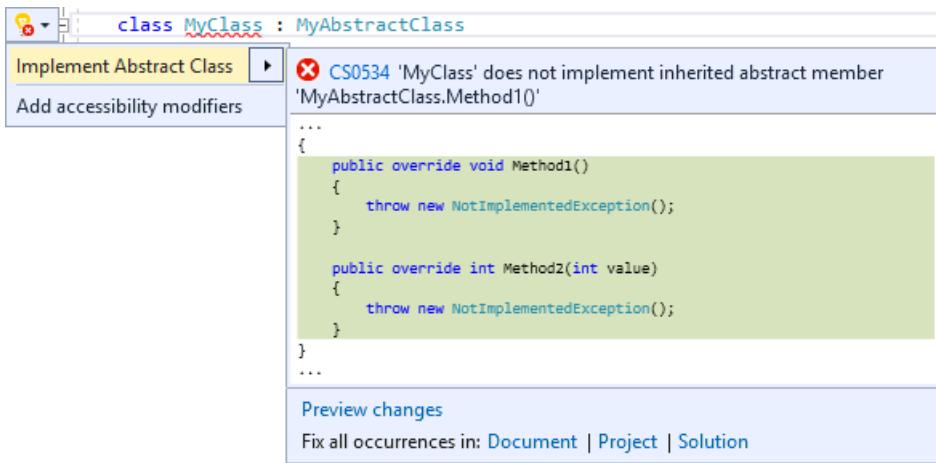
- Visual Basic:

```
Public MustInherit Class MyAbstractClass
    Public MustOverride Sub Method1()
    Public MustOverride Function Method2(value As Integer) As Integer
End Class

Public Class MySubClass
    Inherits MyAbstractClass
End Class
```

2. Next, do one of the following:

- **Keyboard**
 - Press **Ctrl+.** to trigger the **Quick Actions and Refactorings** menu.
- **Mouse**
 - Right-click and select the **Quick Actions and Refactorings** menu.
 - Hover over the red squiggle and click the  icon that appears.
 - Click the  icon that appears in the left margin if the text cursor is already on the line with the red squiggle.



3. Select **Implement Abstract Class** from the drop-down menu.

TIP

- Use the **Preview changes** link at the bottom of the preview window to see all of the changes that will be made before making your selection.
- Use the **Document**, **Project**, and **Solution** links at the bottom of the preview window to create the proper method signatures across multiple classes that inherit from the abstract class.

The abstract method signatures are created, and are ready to be implemented.

- C#:

```
class MyClass : MyAbstractClass
{
    public override void Method1()
    {
        throw new NotImplementedException();
    }

    public override Int32 Method2(Int32 value)
    {
        throw new NotImplementedException();
    }
}
```

- Visual Basic:

```
Public Class MySubClass
    Inherits MyAbstractClass
    Public Overrides Sub Method1()
        Throw New NotImplementedException()
    End Sub

    Public Overrides Function Method2(value As Integer) As Integer
        Throw New NotImplementedException()
    End Function
End Class

End Module
```

See also

- [Code Generation](#)
- [Preview Changes](#)

Implement an interface in Visual Studio

10/18/2019 • 2 minutes to read • [Edit Online](#)

This code generation applies to:

- C#
- Visual Basic

What: Lets you immediately generate the code required to implement an interface.

When: You want to inherit from an interface.

Why: You could manually implement all interface one-by-one, however this feature will generate all method signatures automatically.

How-to

1. Place your cursor on the line where there is a red squiggle that indicates you have referenced an interface, but have not implemented all required members.

- C#:

```
interface IMyInterface
{
    void Method1();
    int Method2(int value);
}

class MyClass : IMyInterface
{
}
```

- Visual Basic:

```
Interface IMyInterface
    Sub Method1()
        Function Method2(value As Integer) As Integer
    End Interface

    Class MyImplementation
        Implements IMyInterface
    End Class
```

2. Next, do one of the following:

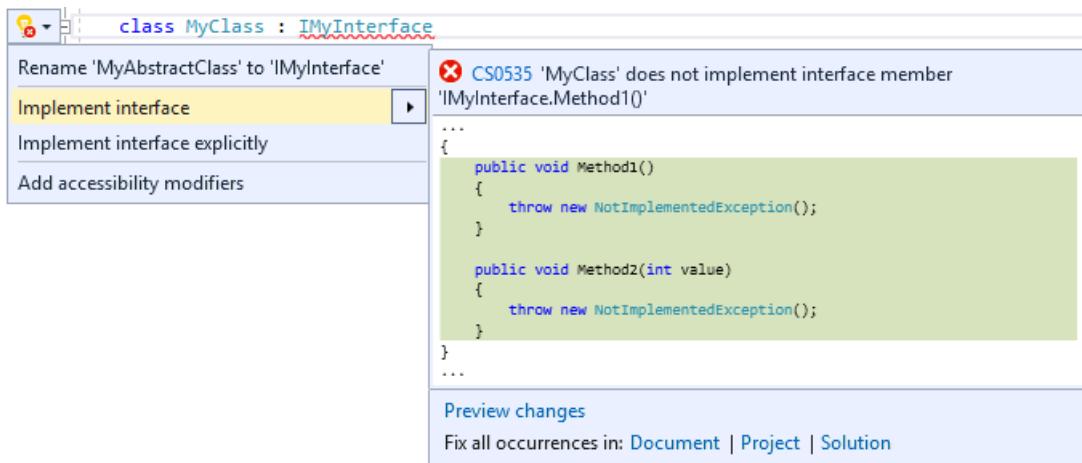
- **Keyboard**

- Press **Ctrl+.** to trigger the **Quick Actions and Refactorings** menu.

- **Mouse**

- Right-click and select the **Quick Actions and Refactorings** menu.
 - Hover over the red squiggle and click the  icon that appears.
 - Click the  icon that appears in the left margin if the text cursor is already on the line with the red squiggle.

3. Select **Implement interface** from the drop-down menu.



TIP

- Use the **Preview changes** link at the bottom of the preview window to see all of the changes that will be made before making your selection.
- Use the **Document**, **Project**, and **Solution** links at the bottom of the preview window to create the proper method signatures across multiple classes that implement the interface.

The interface's method signatures are created, and is ready to be implemented.

- C#:

```
interface IMyInterface
{
    void Method1();
    int Method2(int value);
}

class MyClass : IMyInterface
{
    public void Method1()
    {
        throw new NotImplementedException();
    }

    public Int32 Method2(Int32 value)
    {
        throw new NotImplementedException();
    }
}
```

- Visual Basic:

```
Class MyImplementation
    Implements IMyInterface

    Public Sub Method1() Implements IMyInterface.Method1
        Throw New NotImplementedException()
    End Sub

    Public Function Method2(value As Integer) As Integer Implements IMyInterface.Method2
        Throw New NotImplementedException()
    End Function
End Class
```

TIP

(C# only) Use the **Implement interface explicitly** option to preface each generated method with the interface name to avoid name collisions.

```
interface IMyInterface
{
    void Method1();
    int Method2(int value);
}

class MyClass : IMyInterface
{
    void IMyInterface.Method1()
    {
        throw new NotImplementedException();
    }

    Int32 IMyInterface.Method2(Int32 value)
    {
        throw new NotImplementedException();
    }
}
```

See also

- [Code Generation](#)
- [Preview Changes](#)

Introduce a local variable in Visual Studio

10/18/2019 • 2 minutes to read • [Edit Online](#)

This code generation applies to:

- C#
- Visual Basic

What: Lets you immediately generate a local variable to replace an existing expression.

When: You have code which could be easily reused later if it were in a local variable.

Why: You could copy and paste the code multiple times to use it in various locations, however it would be better to perform the operation once, store the result in a local variable, and use the local variable throughout.

How-to

1. Highlight the expression that you want to assign to a new local variable.

- C#:

```
static void Main(string[] args)
{
    Debug.WriteLine(new Random().Next());
}
```

- Visual Basic:

```
Sub Main()
    Debug.WriteLine(New Random().Next())
End Sub
```

2. Next, do one of the following:

- **Keyboard**

- Press **Ctrl+.** to trigger the **Quick Actions and Refactorings** menu.

- **Mouse**

- Right-click and select the **Quick Actions and Refactorings** menu.

- Click the icon that appears in the left margin if the text cursor is already on the line with the highlighted expression.



3. Select **Introduce local for (all occurrences) of 'expression'** from the drop-down menu.

TIP

Use the **Preview changes** link at the bottom of the preview window to see all of the changes that will be made before making your selection.

The local variable is created, with the type inferred from its usage. Give the new local variable a new name.

- C#:

```
static void Main(string[] args)
{
    Random random = new Random();
    Debug.WriteLine(random.Next());
}
```

- Visual Basic:

```
Sub Main()
    Dim random As Random = New Random()
    Debug.WriteLine(random.Next())
End Sub
```

NOTE

You can use the **...all occurrences of...** menu option to replace every instance of the selected expression, not just the one you have specifically highlighted.

See also

- [Code generation](#)
- [Preview changes](#)

Refactor code

10/18/2019 • 2 minutes to read • [Edit Online](#)

Refactoring is the process of modifying code in order to make it easier to maintain, understand, and extend, but without changing its behavior.

Programming languages

Different refactoring operations are available for different programming languages in Visual Studio:

- The pages in this section of the table of contents cover the refactorings available for C# and Visual Basic. Some examples are [Extract a method refactoring](#) and [Move type to a matching file refactoring](#).
- For information about refactoring C++ code, see [Write and refactor code \(C++\)](#).
- Refactoring support for F# is provided by the [Visual F# Power Tools](#), a third-party Visual Studio extension.

See also

- [Quick Actions](#)
- [Visual Studio IDE](#)
- [Features of the code editor](#)
- [Preview changes](#)
- [Refactoring \(Visual Studio for Mac\)](#)

Change a method signature refactoring

8/1/2019 • 2 minutes to read • [Edit Online](#)

This refactoring applies to:

- C#
- Visual Basic

What: Lets you remove or change the order of a method's parameters.

When: You want to move or remove a method parameter that is currently being used in a variety of locations.

Why: You could manually remove and re-order the parameters, and then find all calls to that method and change them one-by-one, but that could lead to errors. This refactoring tool will perform the task automatically.

How-to

1. Highlight or place the text cursor inside the name of the method to modify, or one of its usages:

- C#:

```
static void Main(string[] args)
{
    ChangeName("John", "Doe");
}

1 reference
private static void ChangeName(string firstName, string lastName)
```

- VB:

```
Sub Main()
    ChangeName("John", "Doe")
End Sub

Sub ChangeName(firstName As String, lastName As String)
End Sub
```

2. Next, do one of the following:

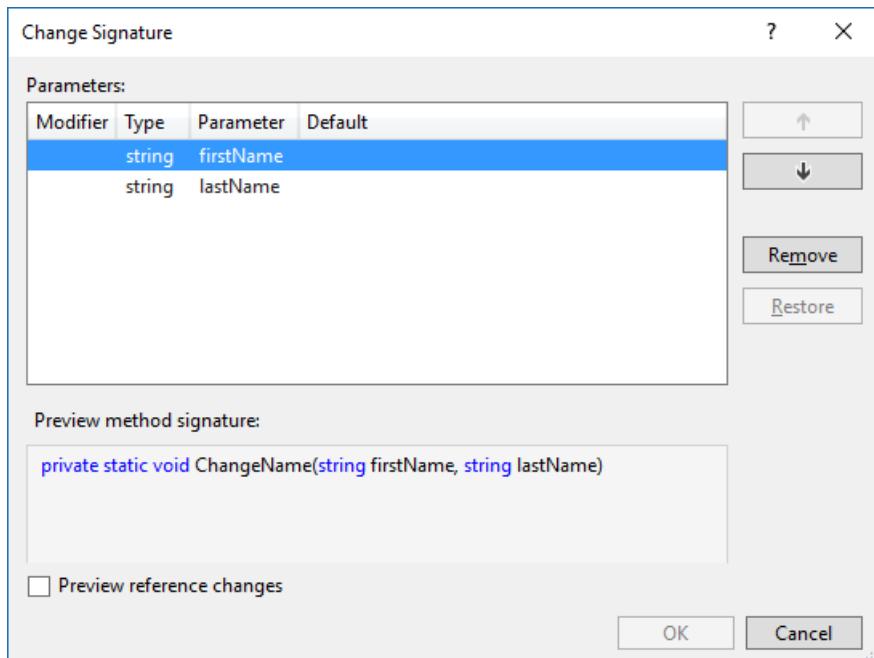
- **Keyboard**

- Press **Ctrl+R**, then **Ctrl+V**. (Note that your keyboard shortcut may be different based on which profile you've selected.)
- Press **Ctrl+.** to trigger the **Quick Actions and Refactorings** menu and select **Change Signature** from the Preview window popup.

- **Mouse**

- Select **Edit > Refactor > Remove Parameters**.
- Select **Edit > Refactor > Reorder Parameters**.
- Right-click the code, select the **Quick Actions and Refactorings** menu and select **Change Signature** from the Preview window popup.

3. In the **Change Signature** dialog that pops up, you can use the buttons on the right side to change the method signature:



BUTTON	DESCRIPTION
Up/Down	Move the selected parameter up and down the list
Remove	Remove the selected parameter from the list
Restore	Restore the selected, crossed-out parameter to the list

TIP

Use the **Preview reference changes** checkbox to see what the result will be before committing to it.

- When you are finished, press the **OK** button to make the changes.

- C#:

```
static void Main(string[] args)
{
    ChangeName("Doe", "John");
}

1 reference
private static void ChangeName(string lastName, string firstName)
```

- Visual Basic:

```
Sub Main()
    ChangeName("Doe", "John")
End Sub

Sub ChangeName(lastName As String, firstName As String)
End Sub
```

See also

- [Refactoring](#)
- [Preview Changes](#)

Convert anonymous type to class

3/18/2019 • 2 minutes to read • [Edit Online](#)

This refactoring applies to:

- C#

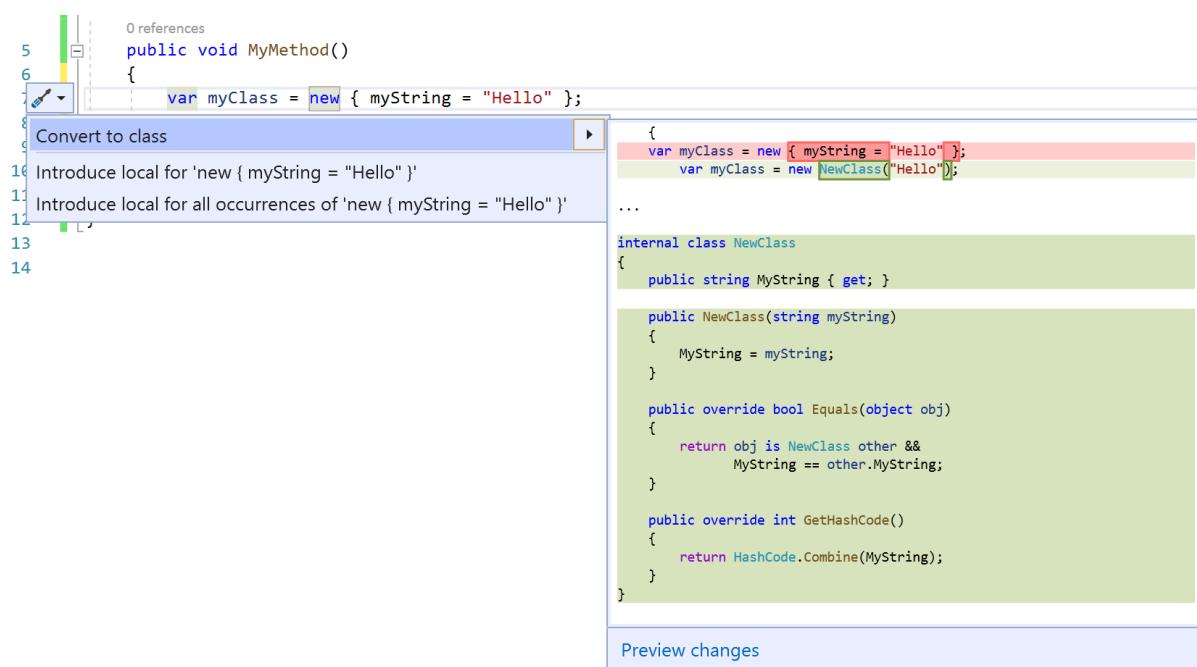
What: Convert an anonymous type to class.

When: You have an anonymous type that you want to continue to build on in a class.

Why: Anonymous types are useful if you're only using them locally. As your code grows, it's nice to have an easy way to promote them to a class.

How-to

1. Place your cursor in an anonymous type.
2. Press **Ctrl+.** to trigger the **Quick Actions and Refactorings** menu.



The screenshot shows a code editor in Visual Studio with the following code:

```
5 0 references
6 public void MyMethod()
{
7     var myClass = new { myString = "Hello" };
8 }
9 Convert to class
10 Introduce local for 'new { myString = "Hello" }'
11 Introduce local for all occurrences of 'new { myString = "Hello" }'
12 ...
13
14 }
```

A context menu is open at line 8, showing the option "Convert to class". Below the code editor, a preview window shows the refactored code:

```
{
    var myClass = new { myString = "Hello" };
    var myClass = new NewClass("Hello");
}

internal class NewClass
{
    public string MyString { get; }

    public NewClass(string myString)
    {
        MyString = myString;
    }

    public override bool Equals(object obj)
    {
        return obj is NewClass other &&
               MyString == other.MyString;
    }

    public override int GetHashCode()
    {
        return HashCode.Combine(MyString);
    }
}
```

At the bottom of the preview window is a button labeled "Preview changes".

3. Press **Enter** to accept the refactoring.

```
0 references
public class Class
{
    0 references
    public void MyMethod()
    {
        var myClass = new NewClass("Hello");

        Console.WriteLine(myClass.MyString);
    }
}

3 references
internal class NewClass
{
    5 references
    public string MyString { get; }

    1 reference
    public NewClass(string myString)
    {
        MyString = myString;
    }

    0 references
    public override bool Equals(object obj)
    {
        return obj is NewClass other &&
               MyString == other.MyString;
    }

    0 references
    public override int GetHashCode()
    {
        return HashCode.Combine(MyString);
    }
}
```

See also

- Refactoring

Convert anonymous type to tuple

3/18/2019 • 2 minutes to read • [Edit Online](#)

This refactoring applies to:

- C#

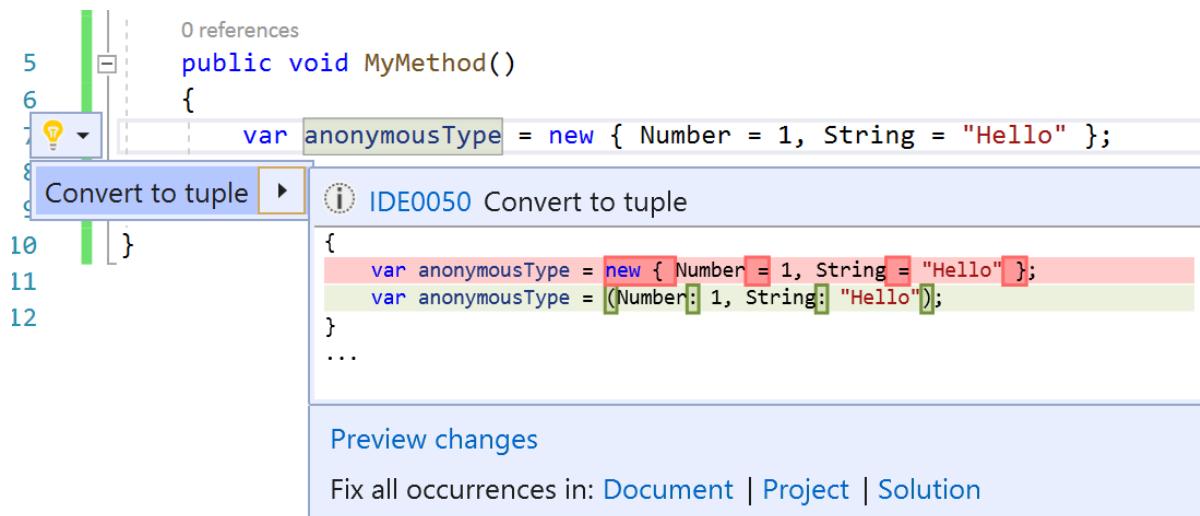
What: Convert an anonymous type to tuple.

When: You have an anonymous type that qualifies as a tuple.

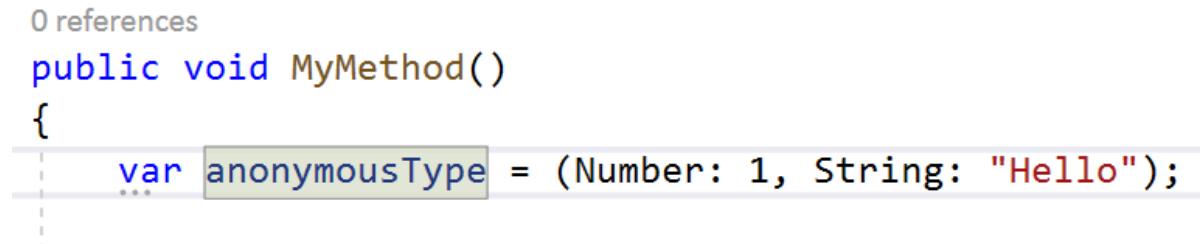
Why: [Tuples](#) are helpful in keeping your syntax lightweight. This quick action makes it easier to take advantage of this C# feature.

How-to

1. Place your cursor in an anonymous type.
2. Press **Ctrl+.** to trigger the **Quick Actions and Refactorings** menu.



3. Press **Enter** to accept the refactoring.



See also

- [Refactoring](#)

Refactoring to convert between a for loop and a foreach statement

10/18/2019 • 2 minutes to read • [Edit Online](#)

This article describes the Quick Actions refactorings that convert between two looping structures. It includes some reasons why you might want to switch between a `for` loop and a `foreach` statement in your code.

Convert a for loop to a foreach statement

If you have a `for` loop in your code, you can use this refactoring to convert it to a `foreach` statement.

This refactoring applies to:

- C#

NOTE

The **Convert to foreach** Quick Action refactoring is only available for `for` loops that contain all three parts: an initializer, condition, and iterator.

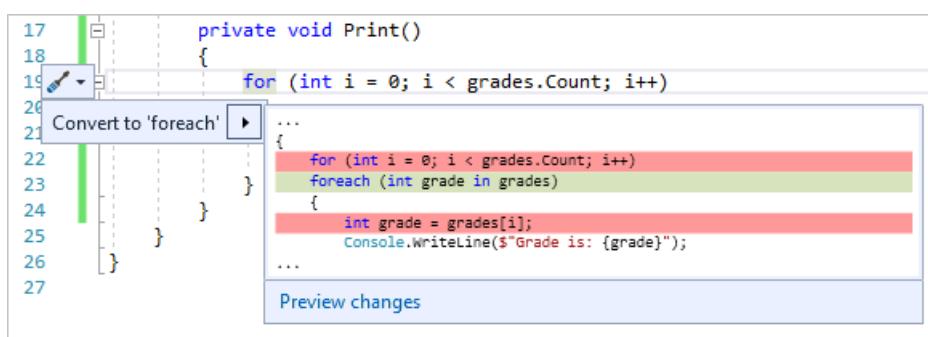
Why convert

Reasons you might want to convert a `for` loop to a `foreach` statement include:

- You don't use the local loop variable inside the loop except as an index to access items.
- You want to simplify your code and reduce the likelihood of logic errors in the initializer, condition, and iterator sections.

How to use it

1. Place your caret in the `for` keyword.
2. Press **Ctrl+.** or click the screwdriver  icon in the margin of the code file.



3. Select **Convert to 'foreach'**. Or, select **Preview changes** to open the **Preview Changes** dialog, and then select **Apply**.

Convert a foreach statement to a for loop

If you have a `foreach` (C#) or `For Each...Next` (Visual Basic) statement in your code, you can use this refactoring to convert it to a `for` loop.

This refactoring applies to:

- C#
- Visual Basic

Why convert

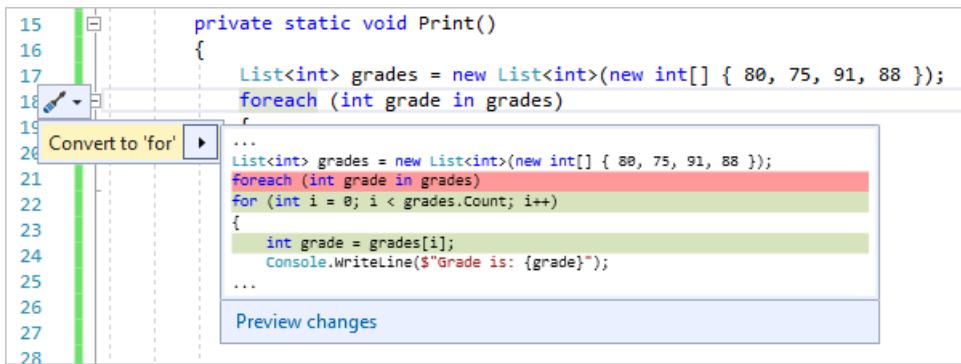
Reasons you might want to convert a `foreach` statement to a `for` loop include:

- You want to use the local loop variable inside the loop for more than just accessing the item.
- You are [iterating through a multi-dimensional array](#) and you want more control over the array elements.

How to use it

1. Place your caret in the `foreach` or `For Each` keyword.

2. Press **Ctrl+.** or click the screwdriver  icon in the margin of the code file.



3. Select **Convert to 'for'**. Or, select **Preview changes** to open the **Preview Changes** dialog, and then select **Apply**.
4. Because the refactoring introduces a new iteration count variable, the **Rename** box appears at the top-right corner of the editor. If you want to choose a different name for the variable, type it in and then press **Enter** or select **Apply** in the **Rename** box. If you don't want to choose a new name, press **Esc** or select **Apply** to dismiss the **Rename** box.

NOTE

For C#, the code generated by these refactorings uses either an explicit type or `var` for the type of the items in the collection. The type in the generated code, explicit or implicit, depends on the code-style settings that are in scope. These particular code-style settings are configured at the machine level under **Tools > Options > Text Editor > C# > Code Style > General > 'var' preferences**, or at the solution level in an [EditorConfig](#) file. If you change a code-style setting in **Options**, reopen the code file for the changes to take effect.

See also

- [Refactoring](#)
- [Preview Changes](#)

Convert Get method to property / Convert property to Get method refactorings

10/18/2019 • 2 minutes to read • [Edit Online](#)

These refactorings apply to:

- C#

Convert Get method to property

What: Lets you convert a Get method into a property (and optionally your Set method).

When: You have a Get method that does not contain any logic.

How-to

1. Place your cursor in your Get method name.

2. Next, do one of the following:

- **Keyboard**
 - Press **Ctrl+.** to trigger the **Quick Actions and Refactorings** menu, and select **Replace method with property** from the Preview window popup.
- **Mouse**
 - Right-click the code, select the **Quick Actions and Refactorings** menu, and select **Replace method with property** from the Preview window popup.

3. (Optional) If you have a Set method, you can also convert your Set method at this time by selecting **Replace Get method and Set method with property**.

4. If you are happy with the change in the code preview, press **Enter** or click the fix from the menu and the changes will be committed.

Example:

```
private int MyValue;

// Before
public int GetMyValue()
{
    return MyValue;
}

// Replace 'GetMyValue' with property

// After
public int MyValue
{
    get { return MyValue; }
}
```

Convert property to Get method

What: Lets you convert a property to a Get method

When: You have a property that involves more than immediately setting and getting a value

How-to

1. Place your cursor in your Get method name.
2. Next, do one of the following:
 - **Keyboard**
 - Press **Ctrl+.** to trigger the **Quick Actions and Refactorings** menu and select **Replace property with methods** from the Preview window popup.
 - **Mouse**
 - Right-click the code, select the **Quick Actions and Refactorings** menu and select **Replace property with methods** from the Preview window popup.
3. If you are happy with the change in the code preview, press **Enter** or click the fix from the menu and the changes will be committed.

See also

- [Refactoring](#)
- [Preview Changes](#)

Convert a local function to a method

9/25/2019 • 2 minutes to read • [Edit Online](#)

This refactoring applies to:

- C#

What: Convert a local function to a method.

When: You have a local function that you want to define outside your current local context.

Why: You want to convert a local function into a method so that you can call it outside your local context. You might want to convert to a method when your local function is getting too long. When you define the function in a separate method, your code is easier to read.

Convert local function to method refactoring

1. Place your cursor in the local function.

```
class Class
{
    0 references
    public void MyMethod()
    {
        string HelloWorld()
        {
            return "H     a string HelloWorld()"
        }
        Console.WriteLine(HelloWorld());
    }
}
```

2. Press **Ctrl+.** to trigger the **Quick Actions and Refactorings** menu.

```
3 class Class
4 {
5     0 references
6     public void MyMethod()
7     {
8         string HelloWorld()
9     }
10 Use expression body for local functions
11 Convert to method
12     }
13 }
14 }
```

The 'Convert to method' option is highlighted in the dropdown menu. A preview window shows the transformed code:

```
string HelloWorld()
{
    return "HelloWorld!";
}
Console.WriteLine(HelloWorld());
```

3. Press Enter to accept the refactoring.

```
class Class
{
    0 references
    public void MyMethod()
    {
        Console.WriteLine(HelloWorld());
    }

    1 reference
    private static string HelloWorld()
    {
        return "HelloWorld!";
    }
}
```

See also

- [Refactoring](#)
- [Tips for .NET Developers](#)

Convert a foreach loop to LINQ

7/24/2019 • 2 minutes to read • [Edit Online](#)

This refactoring applies to:

- C#

What: Lets you easily convert your *foreach* loop that uses an `IEnumerable` to a LINQ query or a LINQ call form (also known as a LINQ method).

When: You have a foreach loop that uses an `IEnumerable`, and you want that loop to read as a LINQ query.

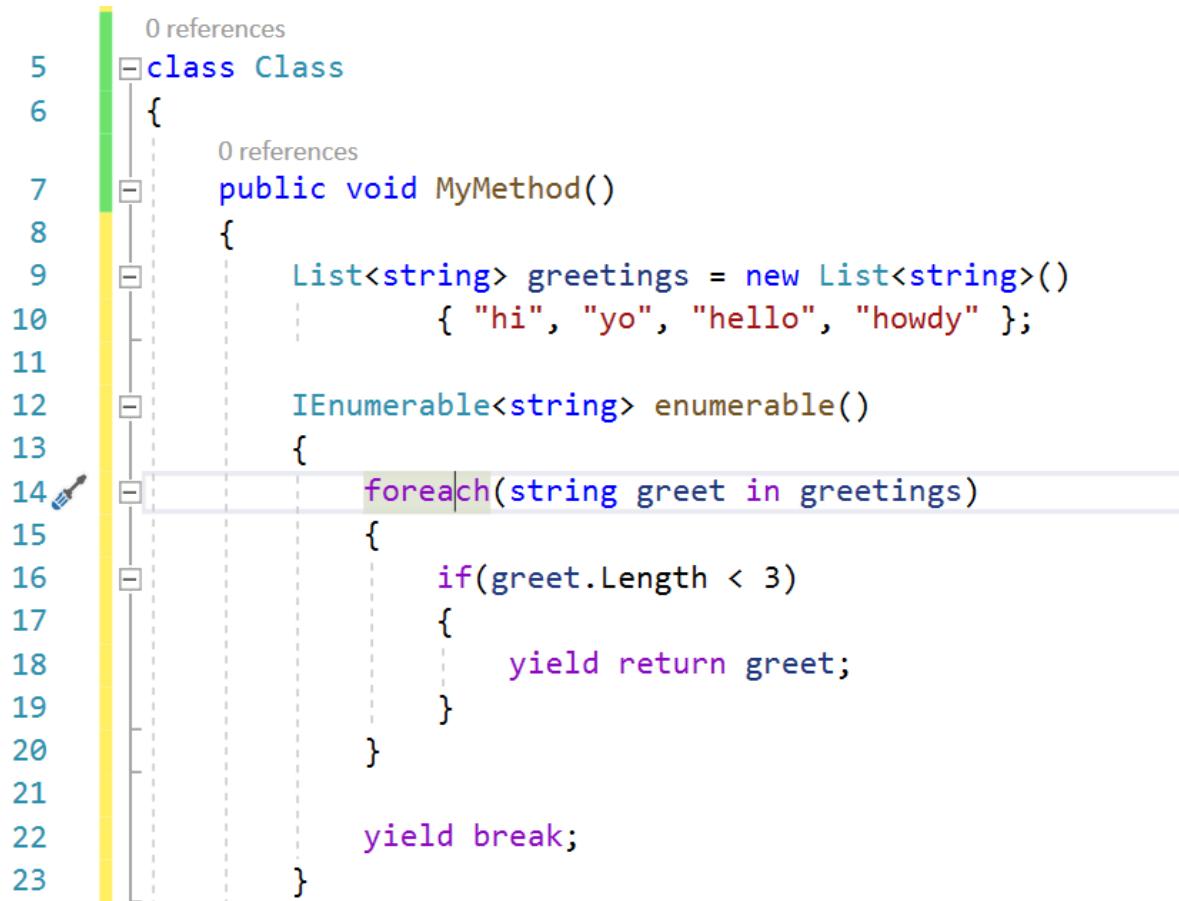
Why: You prefer to use LINQ syntax rather than a foreach loop. LINQ makes a query into a first-class language construct in C#. LINQ can reduce the amount of code in a file, make the code easier to read, and allow different data sources to have similar query expression patterns.

NOTE

LINQ syntax is typically less efficient than a foreach loop. It's good to be aware of any performance tradeoff that might occur when you use LINQ to improve the readability of your code.

Convert a foreach loop to LINQ refactoring

1. Place your cursor in the `foreach` keyword.



```
5 0 references
6 class Class
7 {
8     0 references
9     public void MyMethod()
10    {
11        0 references
12        List<string> greetings = new List<string>()
13            {
14                "hi", "yo", "hello", "howdy"
15            };
16
17        I Enumerable<string> enumerable()
18        {
19            foreach(string greet in greetings)
20            {
21                if(greet.Length < 3)
22                {
23                    yield return greet;
24                }
25            }
26            yield break;
27        }
28    }
29}
```

2. Press **Ctrl + .** to trigger the **Quick Actions and Refactorings** menu.

The screenshot shows a code editor with the following C# code:

```

5 class Class
6 {
7     public void MyMethod()
8     {
9         List<string> greetings = new List<string>()
10        { "hi", "yo", "hello", "howdy" };
11
12         IEnumerable<string> enumerable()
13         {
14             foreach(string greet in greetings)
15         }

```

A context menu is open at the end of the foreach loop, listing three options:

- Convert to LINQ
- Convert to LINQ (call form)
- Convert to 'for'

The 'Convert to LINQ' option is selected, and its preview is shown in a separate window:

```

foreach(string greet in greetings)
{
    if(greet.Length < 3)
    {
        yield return greet;
    }
}

yield break;
return from string greet in greetings
where greet.Length < 3
select greet;
}
...

```

Below the preview is a 'Preview changes' button.

3. Select **Convert to LINQ** or **Convert to Linq (call form)**.

```

0 references
public void MyMethod()
{
    List<string> greetings = new List<string>()
        { "hi", "yo", "hello", "howdy" };

    IEnumerable<string> enumerable()
    {
        return from string greet in greetings
            where greet.Length < 3
            select greet;
    }
}

public void MyMethod()
{
    List<string> greetings = new List<string>()
        { "hi", "yo", "hello", "howdy" };

    IEnumerable<string> enumerable()
    {
        return greetings.Where(greet => greet.Length < 3).Select(greet => greet);
    }
}

```

Sample code

```
using System.Collections.Generic;

public class Class1
{
    public void MyMethod()
    {
        var greetings = new List<string>()
        { "hi", "yo", "hello", "howdy" };

        IEnumerable<string> enumerable()
        {
            foreach (var greet in greetings)
            {
                if (greet.Length < 3)
                {
                    yield return greet;
                }
            }

            yield break;
        }
    }
}
```

See also

- [Refactoring](#)
- [Preview Changes window](#)
- [Tips for .NET Developers](#)

Refactoring to convert LINQ to a foreach statement

10/18/2019 • 2 minutes to read • [Edit Online](#)

Use this refactoring to convert [LINQ query syntax](#) to a [foreach](#) statement.

This refactoring applies to:

- C#

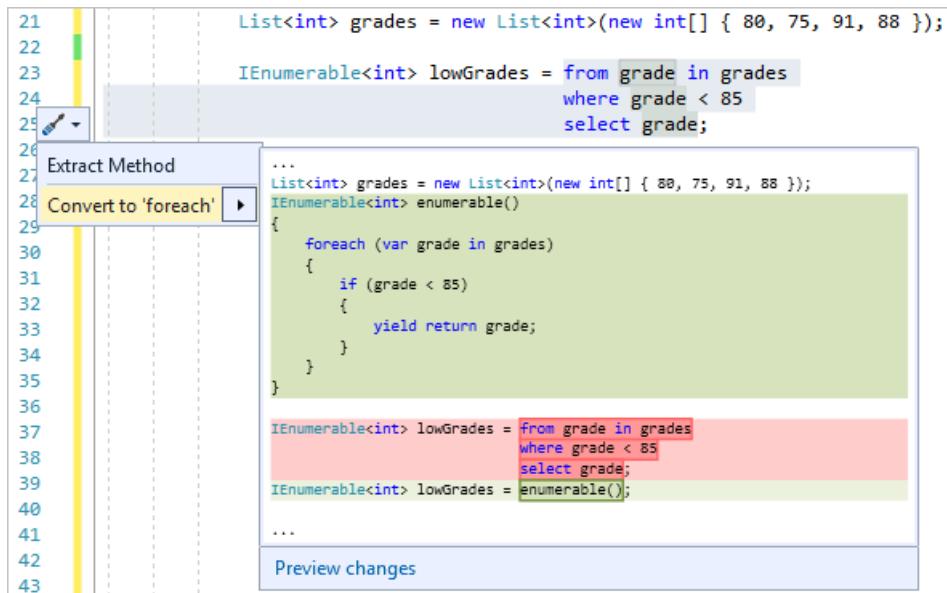
How to use it

1. Select the entire LINQ query starting with `from`.

NOTE

This refactoring can only be used to convert LINQ queries expressed with query syntax and not method syntax.

2. Press **Ctrl+.** or click the screwdriver  icon in the margin of the code file.



The screenshot shows a code editor with the following C# code:

```
21     List<int> grades = new List<int>(new int[] { 80, 75, 91, 88 });
22
23     I Enumerable<int> lowGrades = from grade in grades
24         where grade < 85
25             select grade;
26
27     Extract Method
28     Convert to 'foreach' ▾
29
30     ...
31
32     I Enumerable<int> enumerable()
33     {
34         foreach (var grade in grades)
35         {
36             if (grade < 85)
37             {
38                 yield return grade;
39             }
40         }
41
42     I Enumerable<int> lowGrades = from grade in grades
43         where grade < 85
44             select grade;
45     I Enumerable<int> lowGrades = enumerable();
46
47     ...
48
49     Preview changes
```

A context menu is open at line 28, with the "Convert to 'foreach'" option highlighted. A preview window shows the converted code:

```
...
List<int> grades = new List<int>(new int[] { 80, 75, 91, 88 });
I Enumerable<int> enumerable()
{
    foreach (var grade in grades)
    {
        if (grade < 85)
        {
            yield return grade;
        }
    }
}

I Enumerable<int> lowGrades = from grade in grades
where grade < 85
select grade;
I Enumerable<int> lowGrades = enumerable();

...
Preview changes
```

3. Select **Convert to 'foreach'**. Or, select **Preview changes** to open the **Preview Changes** dialog, and then select **Apply**.

NOTE

For C#, the code generated by these refactorings uses either an explicit type or `var` for the iteration variable of the `foreach` loop. The type in the generated code, explicit or implicit, depends on the code-style settings that are in scope. These particular code-style settings are configured at the machine level under **Tools > Options > Text Editor > C# > Code Style > General > 'var' preferences**, or at the solution level in an [EditorConfig](#) file. If you change a code-style setting in **Options**, reopen the code file for the changes to take effect.

See also

- [LINQ](#)
- [Refactoring](#)

- [Preview Changes](#)

Convert switch statement to switch expression

8/2/2019 • 2 minutes to read • [Edit Online](#)

This refactoring applies to:

- C#

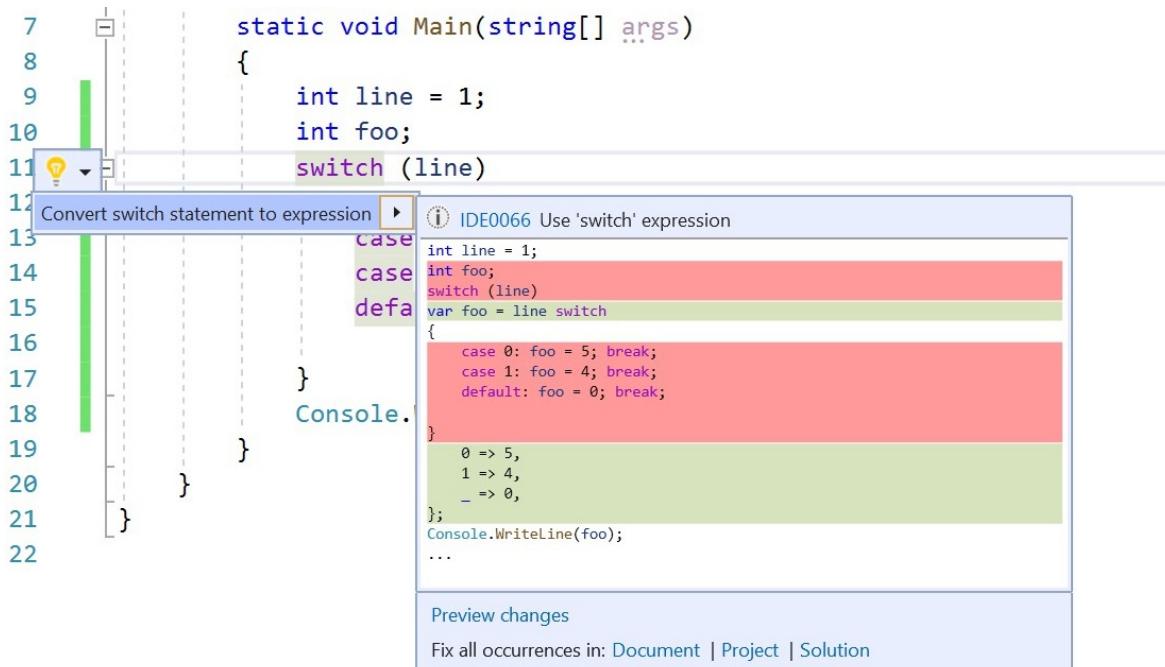
What: Convert a [switch statement](#) to a C# 8.0 [switch expression](#).

When: You want to convert a `switch` statement to a `switch` expression and vice versa.

Why: If you are only using expressions, this refactoring enables an easy transition from traditional `switch` statements.

How-to

1. In your project file, [set the language version to preview](#) since `switch` expressions are a new C# 8.0 feature.
2. Place your cursor in the `switch` keyword and press **Ctrl+.** to trigger the **Quick Actions and Refactorings** menu.
3. Select **Convert switch statement to expression**.



See also

- [Refactoring](#)

Encapsulate a field refactoring

10/18/2019 • 2 minutes to read • [Edit Online](#)

This refactoring applies to:

- C#
- Visual Basic

What: Lets you turn a field into a property, and update all usages of that field to use the newly created property.

When: You want to move a field into a property, and update all references to that field.

Why: You want to give other classes access to a field, but don't want those classes to have direct access. By wrapping the field in a property, you could write code to verify the value being assigned, for example.

How-to

1. Highlight or place the text cursor inside the name of the field to encapsulate:

- C#:

```
class Square
{
    public double side;
}

class Program
{
    static void Main(string[] args)
    {
        Square s = new Square();
        s.side = 1.23;
    }
}
```

- Visual Basic:

```
Class Square
    Public side As Double
End Class

Module Module1
    Sub Main()
        Dim s As New Square
        s.side = 1.23
    End Sub
End Module
```

2. Next, do one of the following:

- **Keyboard**

- Press **Ctrl+R**, then **Ctrl+E**. (Note that your keyboard shortcut may be different based on which profile you've selected.)
- Press **Ctrl+.** to trigger the **Quick Actions and Refactorings** menu and select either **Encapsulate field** entry from the Preview window popup.

- **Mouse**

- Select **Edit > Refactor > Encapsulate Field**.
- Right-click the code, select the **Quick Actions and Refactorings** menu and select either **Encapsulate field** entry from the Preview window popup.

SELECTION	DESCRIPTION
Encapsulate field (and use property)	Encapsulates the field with a property, and updates all usages of the field to use the generated property
Encapsulate field (but still use field)	Encapsulates the field with a property, but leaves all usages of the field untouched

The property is created and references to the field are updated, if selected.

TIP

Use the **Preview changes** link in the popup window to see what the result will be before committing to it.

- C#:

```
class Square
{
    private double side;

    public double Side
    {
        get
        {
            return side;
        }

        set
        {
            side = value;
        }
    }
}

class Program
{
    static void Main(string[] args)
    {
        Square s = new Square();
        s.Side = 1.23;
    }
}
```

- Visual Basic:

```
Class Square
    Private _side As Double

    Public Property Side As Double
        Get
            Return _side
        End Get
        Set(value As Double)
            _side = value
        End Set
    End Property
End Class

Module Module1
    Sub Main()
        Dim s As New Square
        s.Side = 1.23
    End Sub
End Module
```

See also

- [Refactoring](#)

- [Preview Changes](#)

Extract an interface refactoring

10/18/2019 • 2 minutes to read • [Edit Online](#)

This refactoring applies to:

- C#
- Visual Basic

What: Lets you create an interface using existing members from a class, struct, or interface.

When: You have members in a class, struct, or interface that could be inherited by other classes, structs, or interfaces.

Why: Interfaces are great constructs for object-oriented designs. Imagine having classes for various animals (Dog, Cat, Bird) which might all have common methods, such as Eat, Drink, Sleep. Using an interface like IAnimal would allow Dog, Cat, and Bird to have a common "signature" for these methods.

Extract an interface refactoring

1. Place your cursor in the class name.

- C#:

```
class Dog
{
    public void Eat()
    {
    }

    public void Drink(int value)
    {
    }

    public int Sleep()
    {
        return 0;
    }
}
```

- Visual Basic:

```
Class Dog
    Public Sub Eat()

    End Sub

    Public Sub Drink(value As Integer)

    End Sub

    Public Function Sleep() As Integer
        Return 0
    End Function
End Class
```

2. Next, do one of the following actions:

- **Keyboard**

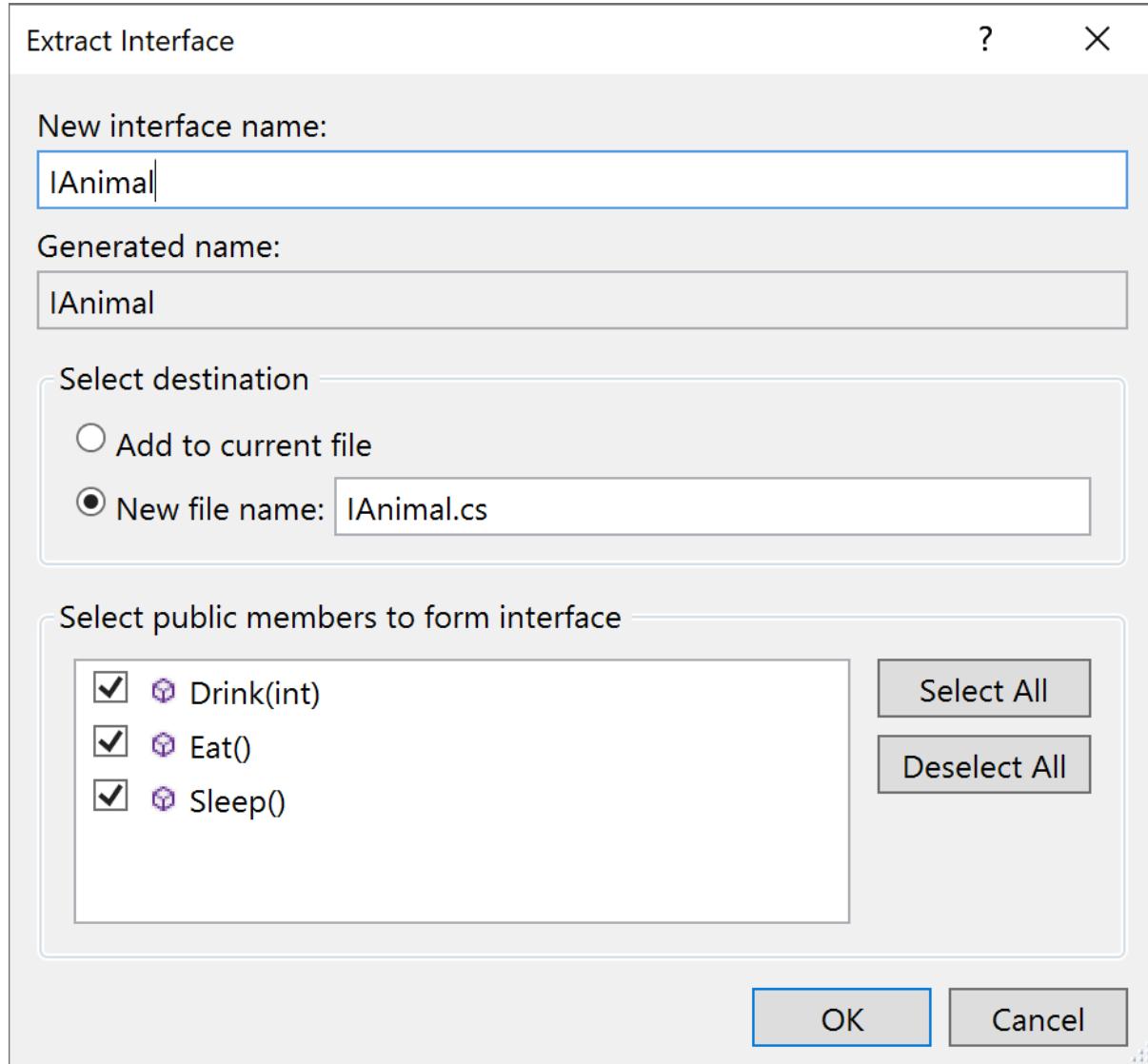
- Press **Ctrl+R**, then **Ctrl+I**. (Your keyboard shortcut may be different based on which profile you've selected.)
- Press **Ctrl+.** to trigger the **Quick Actions and Refactorings** menu and select **Extract Interface**

from the Preview window popup.

- **Mouse**

- Select **Edit > Refactor > Extract Interface**.
- Right-click the name of the class, select the **Quick Actions and Refactorings** menu and select **Extract Interface** from the Preview window popup.

3. In the **Extract Interface** dialog box that pops up, enter the information asked:



FIELD	DESCRIPTION
New interface name	The name of the interface to be created. The name will default to <code>IClassName</code> , where <i>ClassName</i> is the name of the class you selected above.
New file name	The name of the generated file that will contain the interface. As with the interface name, this name will default to <code>IClassName.cs</code> , where <i>ClassName</i> is the name of the class you selected above. You can also select the option to Add to current file .
Select public members to form interface	The items to extract into the interface. You may select as many as you wish.

4. Choose **OK**.

The interface is created in the file of the name specified. Additionally, the class you selected implements that

interface.

- C#:

```
class Dog : IAnimal
{
    public void Eat()
    {
    }

    public void Drink(int value)
    {
    }

    public int Sleep()
    {
        return 0;
    }
}

interface IAnimal
{
    void Drink(int value);
    void Eat();
    int Sleep();
}
```

- Visual Basic:

```
Class Dog
    Implements IAnimal

    Public Sub Eat() Implements IAnimal.Eat
    End Sub

    Public Sub Drink(value As Integer) Implements IAnimal.Drink
    End Sub

    Public Function Sleep() As Integer Implements IAnimal.Sleep
        Return 0
    End Function
End Class

Interface IAnimal
    Sub Drink(value As Integer)
    Sub Eat()
    Function Sleep() As Integer
End Interface
```

See also

- [Refactoring](#)
- [Tips for .NET Developers](#)

Extract a method refactoring

10/18/2019 • 2 minutes to read • [Edit Online](#)

This refactoring applies to:

- C#
- Visual Basic

What: Lets you turn a fragment of code into its own method.

When: You have a fragment of existing code in some method that needs to be called from another method.

Why: You could copy/paste that code, but that would lead to duplication. A better solution is to refactor that fragment into its own method which can be called freely by any other method.

How-to

1. Highlight the code to be extracted:

- C#:

```
class Program
{
    static void Main(string[] args)
    {
        double radius = 1.23;

        double area = Math.PI * radius * radius;
    }
}
```

- Visual Basic:

```
Sub Main()
    Dim radius As Double

    Dim area As Double = Math.PI * radius * radius
End Sub
```

2. Next, do one of the following:

- **Keyboard**

- Press **Ctrl+R**, then **Ctrl+M**. (Note that your keyboard shortcut may be different based on which profile you've selected.)
- Press **Ctrl+.** to trigger the **Quick Actions and Refactorings** menu and select **Extract Method** from the Preview window popup.

- **Mouse**

- Select **Edit > Refactor > Extract Method**.
- Right-click the code and select **Refactor > Extract > Extract Method**.
- Right-click the code, select the **Quick Actions and Refactorings** menu and select **Extract Method** from the Preview window popup.

The method will be immediately created. From here, you can now rename the method simply by typing the new name.

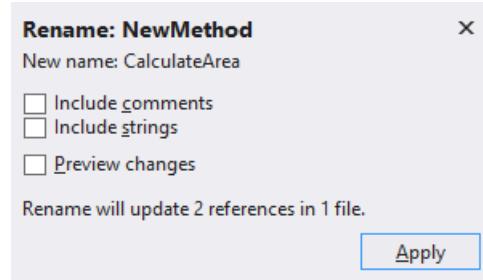
TIP

You can also update comments and other strings to use this new name, as well as preview changes before saving, using the checkboxes in the **Rename** box that appears at the top right of your IDE.

- C#:

```
using System;
namespace ConsoleApplication1
{
    class Program
    {
        static void Main(string[] args)
        {
            double radius = 1.23;
            CalculateArea(radius);
        }

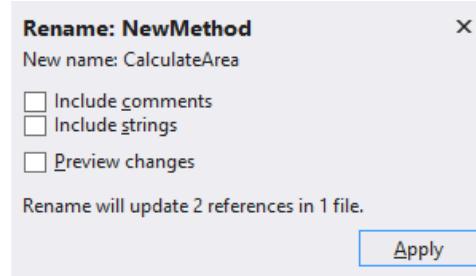
        private static void CalculateArea(Double radius)
        {
            double area = Math.PI * radius * radius;
        }
    }
}
```



- Visual Basic:

```
Module Module1
    Sub Main()
        Dim radius As Double
        CalculateArea(radius)
    End Sub

    Private Sub CalculateArea(radius As Double)
        Dim area As Double = Math.PI * radius * radius
    End Sub
End Module
```



3. When you're happy with the change, choose the **Apply** button or press **Enter** and the changes will be committed.

See also

- [Refactoring](#)
- [Preview Changes](#)

Generate parameter

6/22/2019 • 2 minutes to read • [Edit Online](#)

This refactoring applies to:

- C#

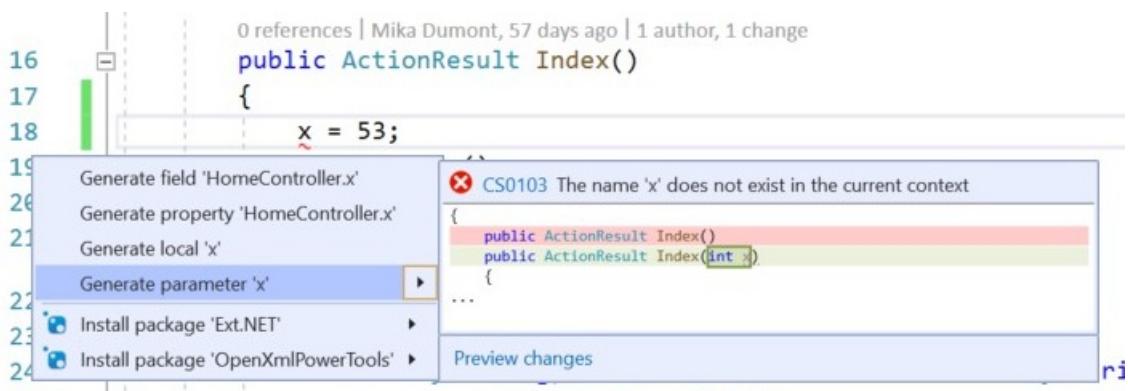
What: Automatically generates a method parameter.

When: You reference a variable in a method that doesn't exist in the current context and receive an error; you can generate a parameter as a code fix.

Why: You can quickly modify a method signature without losing context.

How-to

1. Place your cursor in the variable name and press **Ctrl+.** to trigger the **Quick Actions and Refactorings** menu.
2. Select **Generate parameter**.



See also

- [Refactoring](#)

Inline a temporary variable refactoring

10/18/2019 • 2 minutes to read • [Edit Online](#)

This refactoring applies to:

- C#
- Visual Basic

What: Lets you remove a temporary variable and replace it with its value instead.

When: The use of the temporary variable makes the code harder to understand.

Why: Removing a temporary variable may make the code easier to read.

How-to

1. Highlight or place the text cursor inside the temporary variable to be inlined:

- C#:

```
static void Main(string[] args)
{
    double radiusSquared = 1.23 * 1.23;
    double area = Math.PI * radiusSquared;
}
```

- Visual Basic:

```
Dim radiusSquared As Double = 1.23 * 1.23
Dim area = Math.PI * radiusSquared
```

2. Next, do one of the following:

- **Keyboard**

- Press **Ctrl+.** to trigger the **Quick Actions and Refactorings** menu.

- **Mouse**

- Right-click the code and select the **Quick Actions and Refactorings** menu.

3. Select **Inline temporary variable** from the Preview window popup.

The variable is removed and its usages replaced by the value of the variable.

- C#:

```
static void Main(string[] args)
{
    double area = Math.PI * 1.23 * 1.23;
}
```

- Visual Basic:

```
Dim area = Math.PI * 1.23 * 1.23
```

See also

- [Refactoring](#)

IntelliSense completion for unimported types

6/21/2019 • 2 minutes to read • [Edit Online](#)

This refactoring applies to:

- C#

What: IntelliSense gives completion for unimported types.

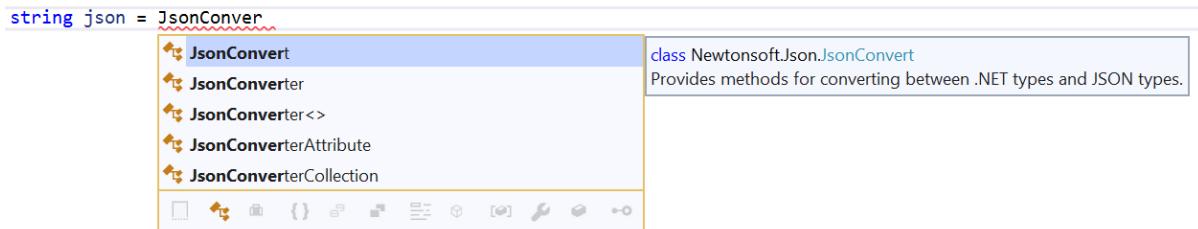
When: You want to add a type that already has a dependency in your project but the import statement has not yet been added to your file.

Why: You don't have to manually add the import statement to your file.

How-to

1. Once you start using a type that has a dependency in your project, IntelliSense will give you suggestions.
2. Press **Tab**.

The import statement will be added to your file.



See also

- [Refactoring](#)

Invert conditional expressions and conditional AND/OR operators

5/10/2019 • 2 minutes to read • [Edit Online](#)

This refactoring applies to:

- C#
- Visual Basic

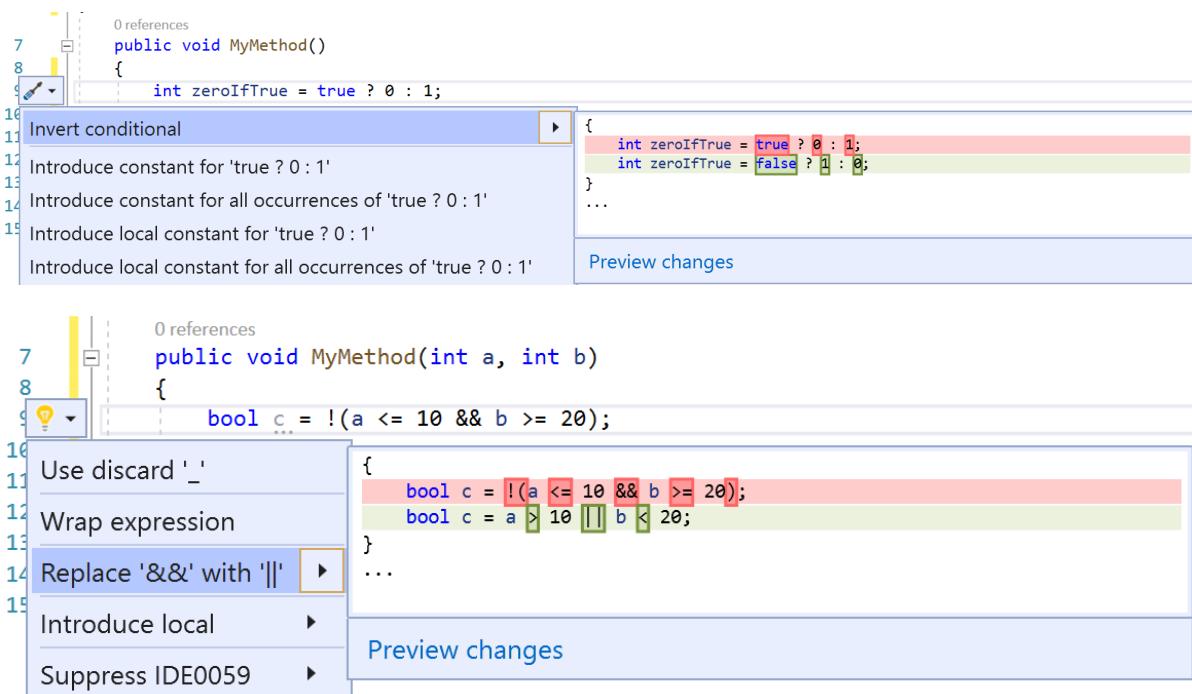
What: Lets you invert a conditional expression or a conditional AND/OR operator.

When: You have a conditional expression or conditional AND/OR operator that would be better understood if inverted.

Why: Inverting an expression or conditional AND/OR operator by hand can take much longer and possibly introduce errors. This code fix helps you do this refactoring automatically.

Invert conditional expressions and conditional AND/OR operators refactoring

1. Place your cursor in a conditional expression or a conditional AND/OR operator.
2. Press **Ctrl+.** to trigger the **Quick Actions and Refactorings** menu.
3. Select **Invert conditional** or **Replace '&&' with '||'**



See also

- [Refactoring](#)
- [Tips for .NET Developers](#)

Invert if statement

5/10/2019 • 2 minutes to read • [Edit Online](#)

This refactoring applies to:

- C#
- Visual Basic

What: Lets you invert an `if` or `if else` statement without changing the meaning of the code.

When: When you have an `if` or `if else` statement that would be better understood when inverted.

Why: Inverting an `if` or `if else` statement by hand can take much longer and possibly introduce errors. This code fix helps you do this refactoring automatically.

Invert if statement refactoring

1. Place your cursor in an `if` or `if else` statement.



A screenshot of a code editor showing a C# class named `Class`. The code contains a `MyMethod()` method with the following content:

```
class Class
{
    public void MyMethod()
    {
        if(1 > 2)
        {
            Console.WriteLine("Change places!");
        }
        else
        {
            Console.WriteLine("Okay!");
        }
    }
}
```

The `if` statement is highlighted with a yellow background, and the condition `1 > 2` is underlined with a red squiggle. The code editor's status bar at the bottom shows "0 references".

2. Press **Ctrl+.** to trigger the **Quick Actions and Refactorings** menu.

The screenshot shows a code editor with the following C# code:

```
5 class Class
6 {
7     public void MyMethod()
8     {
9         if(1 > 2)
10    {
11        if(1 > 2)
12        if(1 <= 2)
13        {
14            Console.WriteLine("Change places!");
15            Console.WriteLine("Okay!");
16        }
17        else
18        {
19            Console.WriteLine("Okay!");
20            Console.WriteLine("Change places!");
21        }
22    }
23 }
24
```

A context menu is open at the end of the first line of the inner if-block (line 11). The "Invert if" option is highlighted. A preview window titled "Preview changes" shows the code after the refactoring:

```
11 Invert if ▶
12 if(1 <= 2)
13 if(1 > 2)
14 {
15     Console.WriteLine("Change places!");
16     Console.WriteLine("Okay!");
17 }
18 else
19 {
20     Console.WriteLine("Okay!");
21     Console.WriteLine("Change places!");
22 }
```

The "Preview changes" button is visible at the bottom of the dialog.

3. Select **Invert if**.

The screenshot shows the code after the "Invert if" refactoring has been applied. The inner if-block has been swapped:

```
class Class
{
    public void MyMethod()
    {
        if(1 <= 2)
        {
            Console.WriteLine("Okay!");
        }
        else
        {
            Console.WriteLine("Change places!");
        }
    }
}
```

See also

- [Refactoring](#)
- [Tips for .NET Developers](#)

Move declaration near reference refactoring

10/18/2019 • 2 minutes to read • [Edit Online](#)

This refactoring applies to:

- C#

What: Lets you move variable declarations closer to their usage.

When: You have variable declarations that can be in a narrower scope.

Why: You could leave it as it is, but that may cause readability issues or information hiding. This is a chance to refactor to improve readability.

How-to

1. Place your cursor in the variable declaration.
2. Next, do one of the following:
 - **Keyboard**
 - Press **Ctrl+.** to trigger the **Quick Actions and Refactorings** menu and select **Move declaration near reference** from the Preview window popup.
 - **Mouse**
 - Right-click the code, select the **Quick Actions and Refactorings** menu and select **Move declaration near reference** from the Preview window popup.
3. When you're happy with the change, press **Enter** or click the fix in the menu and the changes will be committed.

Example:

```
// Before
int x;
if (condition)
{
    x = 1;
    Console.WriteLine(x);
}

// Move declaration near reference

// After
if (condition)
{
    int x = 1;
    Console.WriteLine(x);
}
```

See also

- [Refactoring](#)
- [Preview Changes](#)

Move a type to a matching file refactoring

10/18/2019 • 2 minutes to read • [Edit Online](#)

This refactoring applies to:

- C#
- Visual Basic

What: Lets you move the selected type to a separate file with the same name.

When: You have multiple classes, structs, interfaces, etc. in the same file which you want to separate.

Why: Placing multiple types in the same file can make it difficult to find these types. By moving types to files with the same name, code becomes more readable and easier to navigate.

How-to

1. Place the cursor inside the name of the type where it is defined. For example:

```
class Person
```

```
Class Person
```

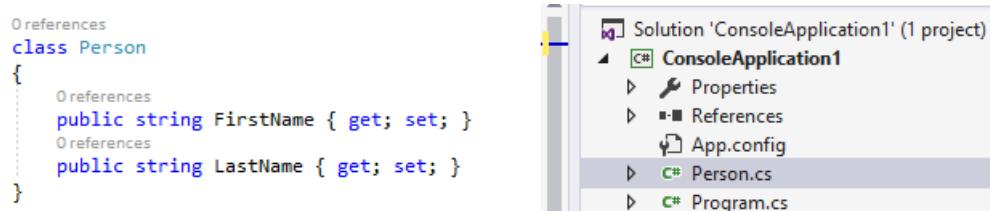
2. Next, do one of the following:

- Press **Ctrl+.**
- Right-click on the type name and select **Quick Actions and Refactorings**

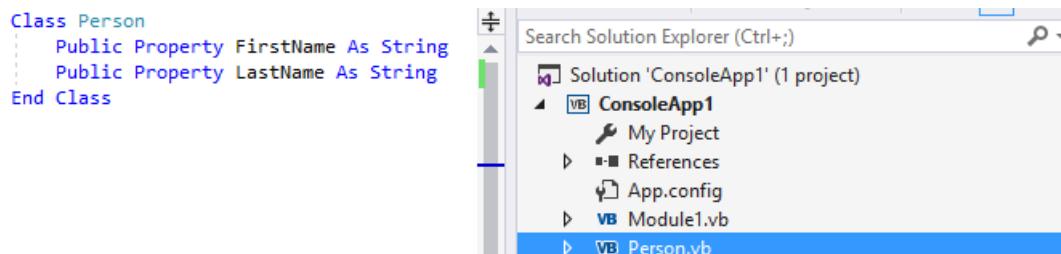
3. Select **Move type to TypeName.cs** from the menu, where *TypeName* is the name of the type you've selected.

The type is moved to a new file in the project that has the same name as the type.

- C#:



- Visual Basic:



See also

- [Refactoring](#)

Move type to namespace

6/20/2019 • 2 minutes to read • [Edit Online](#)

This refactoring applies to:

- C#

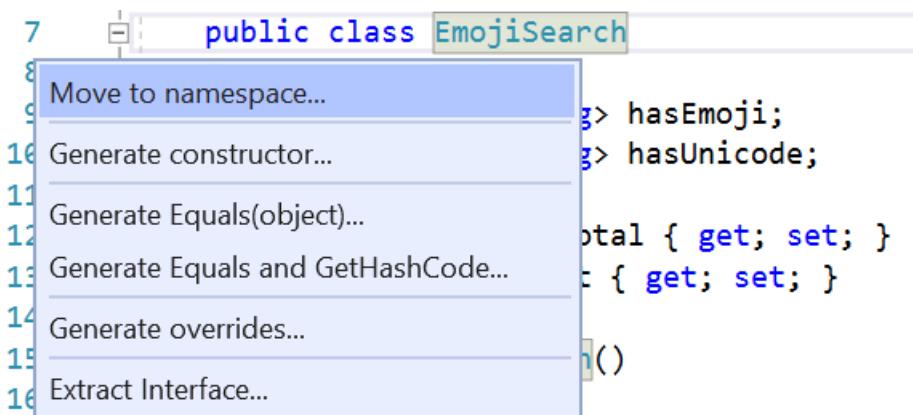
What: Move type to namespace.

When: You want to move a type to a different namespace or folder.

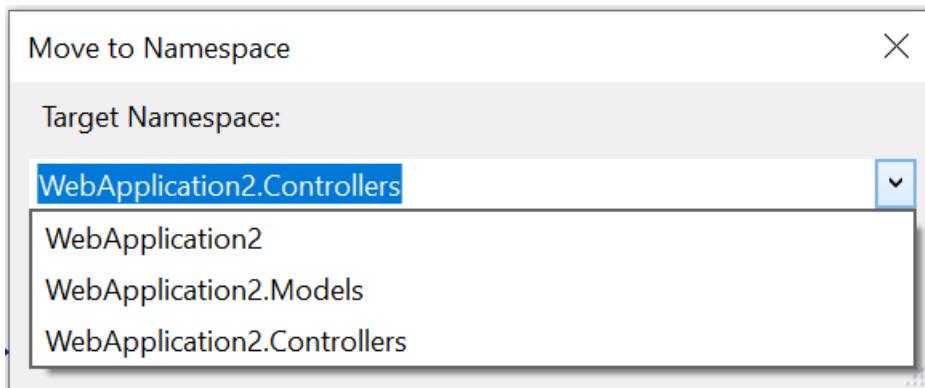
Why: You want to refactor parts of your solution and have a quick way to move a type to a different namespace or folder.

How-to

1. Place your cursor in the class name.
2. Press **Ctrl+.** to trigger the **Quick Actions and Refactorings** menu.
3. Select **Move to namespace**.



4. In the dialog box that opens, select the target namespace you'd like to move the type to.



See also

- [Refactoring](#)

Pull members up

3/18/2019 • 2 minutes to read • [Edit Online](#)

This refactoring applies to:

- C#
- Visual Basic

What: Lets you pull members up to the base type.

When: You have implemented an interface and you want to move a member to the base type.

Why: Pulling members up enables other implementations of your interface to inherit those members as well.

How-to

1. Place your cursor in any member of an implemented interface.
2. Press **Ctrl+.** to trigger the **Quick Actions and Refactorings** menu.

The screenshot shows a code editor with the following code:

```
2     public interface MyInterface
3     {
4     }
5     public class MyClass : MyInterface
6     {
7         public void PullUpMethod() { }
8     }

```

A vertical toolbar icon is visible on the left. The 'PullUpMethod()' line is highlighted with a yellow background. A tooltip at the bottom of the screen says:

Pull 'MyClass.PullUpMethod()' up to 'MyInterface'
Pull members up to base type...

3. Select **Pull Members up to base type.**
4. In the dialog, select what members you would like to add to the selected interface.

Select destination and members to pull up.

Select destination:

MyInterface

Select members:

Members

Make abstract

Select Dependents

PullUpMethod()

Select Public

OK

Cancel

5. Choose **OK**. The selected members are pulled up to the interface.

```
public interface MyInterface
{
    1 reference
    void PullUpMethod();
}

0 references
public class MyClass : MyInterface
{
    1 reference
    public void PullUpMethod() { }
}
```

See also

- [Refactoring](#)

Regex completion through IntelliSense menu

6/12/2019 • 2 minutes to read • [Edit Online](#)

This refactoring applies to:

- C#

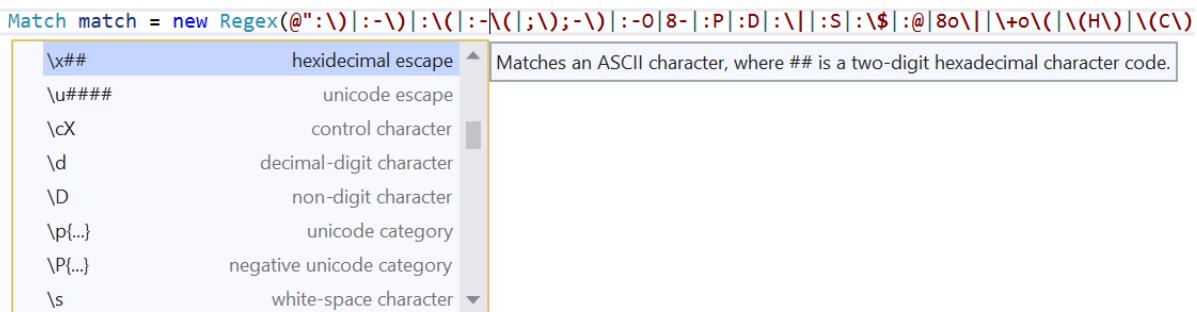
What: Regular expression (regex) completion through IntelliSense menu.

When: You want to write a regular expression with help from IntelliSense. IntelliSense gives you basic completion and an explanation as to what each of the regex characters mean.

Why: Writing regex is hard and IntelliSense can help you write it.

How-to

1. Place your cursor in the regex string.
2. Press **Ctrl+Space** to trigger the **IntelliSense** menu.
3. Select the character you would like to add to your regex string.



See also

- [Refactoring](#)

Remove unreachable code refactoring

10/18/2019 • 2 minutes to read • [Edit Online](#)

This refactoring applies to:

- C#

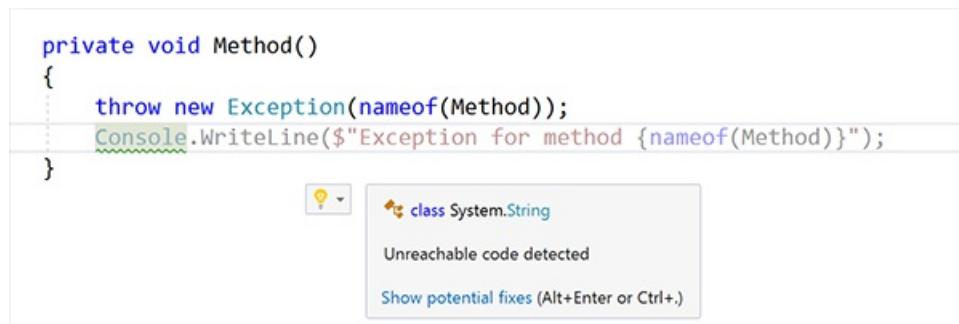
What: Removes code that will never be executed.

When: Your program has no path to a code snippet, making that code snippet unnecessary.

Why: Improve readability and maintainability by removing code that is superfluous and will never be executed.

How-to

1. Place your cursor anywhere in the faded out code that is unreachable:



A screenshot of a code editor showing a tooltip for unreachable code. The tooltip is titled 'Unreachable code detected' and contains a link 'Show potential fixes (Alt+Enter or Ctrl+.)'. The code in the editor is:

```
private void Method()
{
    throw new Exception(nameof(Method));
    Console.WriteLine($"Exception for method {nameof(Method)}");
}
```

1. Next, do one of the following:

- **Keyboard**

- Press **Ctrl+.** to trigger the **Quick Actions and Refactorings** menu and select **Remove unreachable code** from the Preview window popup.

- **Mouse**

- Right-click the code, select the **Quick Actions and Refactorings** menu and select **Remove unreachable code** from the Preview window popup.

2. When you're happy with the change, press **Enter** or click the fix in the menu and the changes will be committed.

Example:

```
// Before
private void Method()
{
    throw new Exception(nameof(Method));
    Console.WriteLine($"Exception for method {nameof(Method)}");
}

// Remove unreachable code

// After
private void Method()
{
    throw new Exception(nameof(Method));
}
```

See also

- [Refactoring](#)
- [Preview Changes](#)

Rename a code symbol refactoring

10/18/2019 • 2 minutes to read • [Edit Online](#)

This refactoring applies to:

- C#
- Visual Basic

What: Lets you rename identifiers for code symbols, such as fields, local variables, methods, namespaces, properties and types.

When: You want to safely rename something without having to find all instances, and copy/paste the new name.

Why: Copy and pasting the new name across an entire project would likely result in errors. This refactoring tool will accurately perform the renaming action.

How-to

1. Highlight or place the text cursor inside the item to be renamed:

- C#:

```
static void Main(string[] args)
{
    double r = 1.23;

    double area = Math.PI * r * r;
}
```

- Visual Basic:

```
Dim r As Double = 1.23

Dim area As Double = Math.PI * r * r
```

2. Next, do one of the following:

- **Keyboard**

- Press **Ctrl+R**, then **Ctrl+R**. (Note that your keyboard shortcut may be different based on which profile you've selected.)

- **Mouse**

- Select **Edit > Refactor > Rename**.
 - Right-click the code and select **Rename**.

3. Rename the item simply by typing the new name.

- C#:

```

using System;

namespace ConsoleApplication1
{
    class Program
    {
        static void Main(string[] args)
        {
            double r = 1.23;

            double area = Math.PI * r * r;
        }
    }
}

```

- Visual Basic:

```

Dim radius As Double = 1.23

Dim area As Double = Math.PI * radius * radius

```

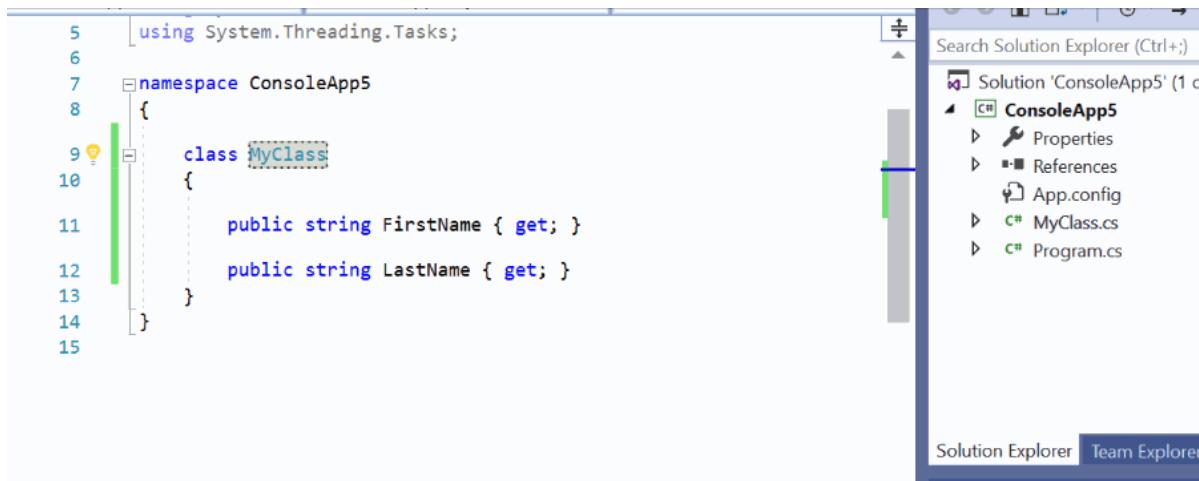
TIP

You can also update comments and other strings to use this new name, as well as [preview the changes](#) before saving, using the checkboxes in the **Rename** box that appears at the top right of your editor.

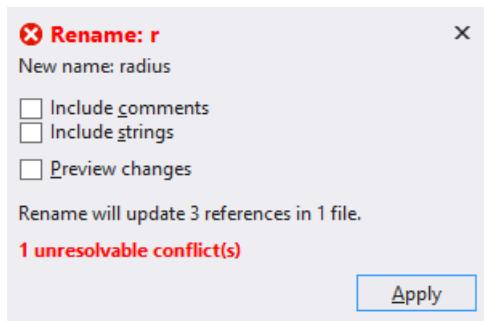
4. When you're happy with the change, choose the **Apply** button or press **Enter** and the changes will be committed.

Remarks

- Starting in Visual Studio 2019 version 16.3, when you rename a type that matches the name of the file it's in, a checkbox appears that enable you to rename the file at the same time. This option appears when you rename a class, interface, or enumeration. This option is not supported for partial types with multiple definitions.



- If you use a name that already exists which would cause a conflict, the **Rename** box will warn you.



- Another way to rename a symbol is to change its name in the editor. Then, with the cursor in the symbol name, press **Ctrl+.** or just expand the light bulb icon menu that appears and choose **Rename <old name> to <new name>**.

See also

- [Refactoring](#)
- [Preview Changes](#)

Sort usings

6/22/2019 • 2 minutes to read • [Edit Online](#)

This refactoring applies to:

- C#

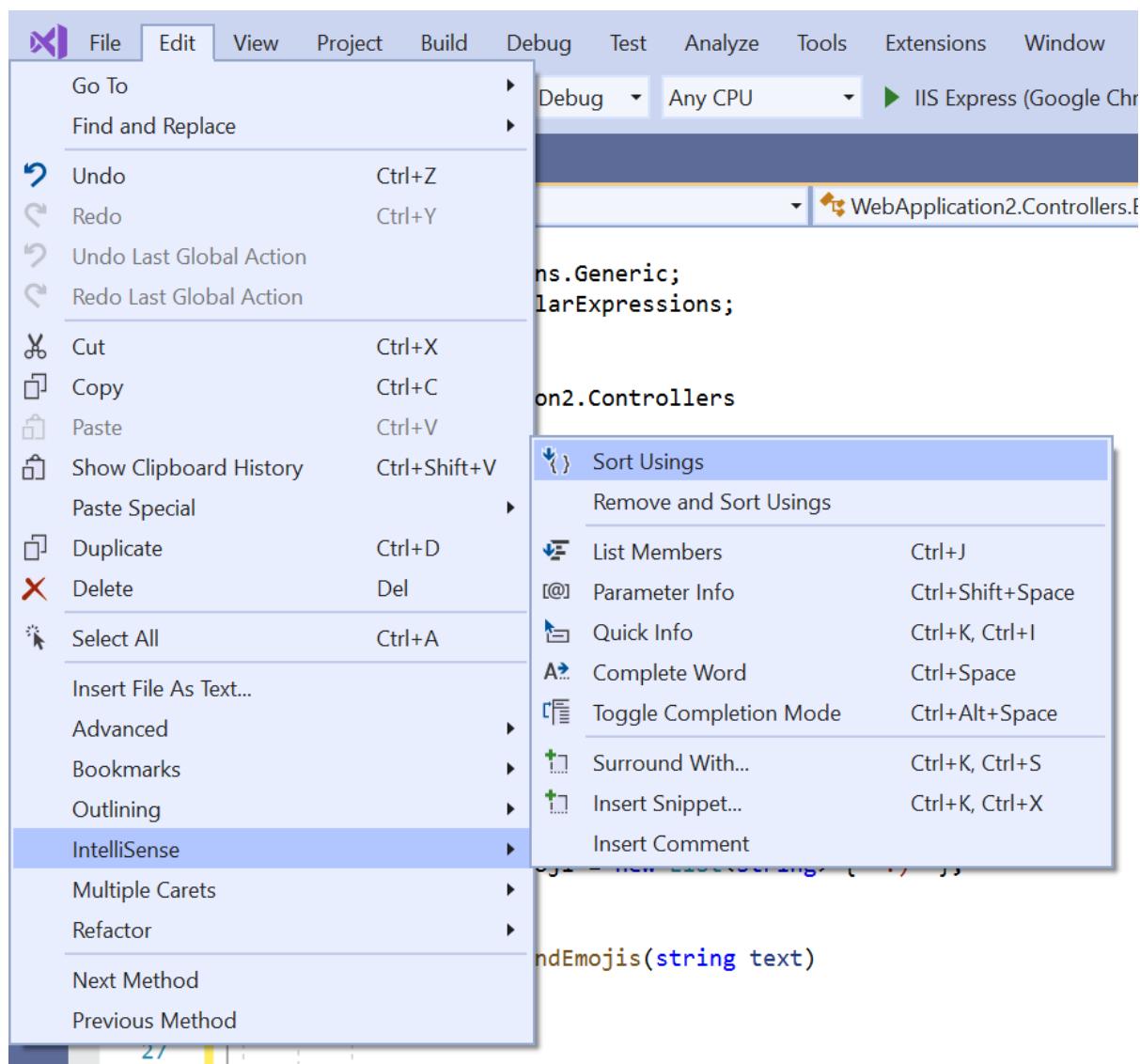
What: Sort usings.

When: You want to sort `using` directives at the top of your file so that they are in alphabetical order.

Why: It makes it easier to find a using directive.

How-to

1. Select **Edit** from the menu bar.
2. Select **Intellisense > Sort Usings**.



See also

- [Refactoring](#)

Split or merge if statements

6/17/2019 • 2 minutes to read • [Edit Online](#)

This refactoring applies to:

- C#

What: What: Split or merge if statements.

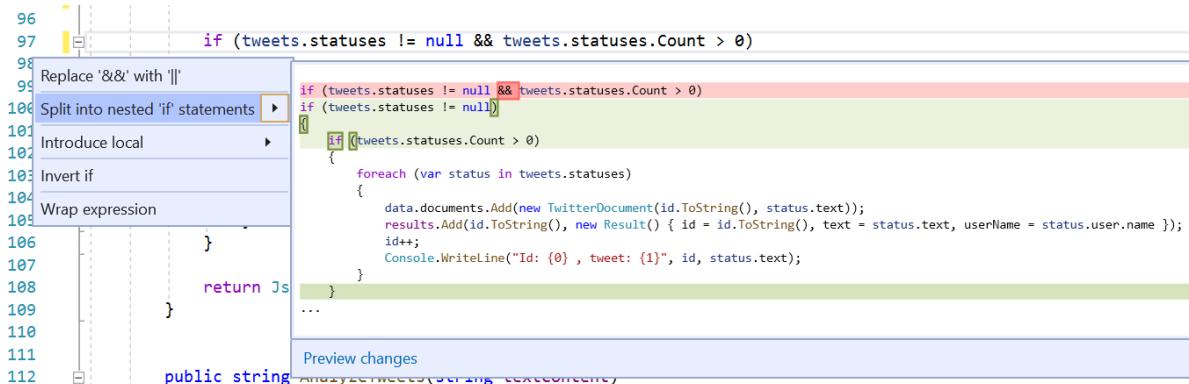
When: You want to split an if statement that uses the && or || operators into a nested if statement, or merge an if statement with an outer if statement.

Why: It's a matter of style preference.

How-to

If you want to split the if statement:

1. Place your cursor in the if statement by the && or || operator.
2. Press **Ctrl+.** to trigger the **Quick Actions and Refactorings** menu.



3. Select **Split into nested if statements**.

```
if (tweets.statuses != null)
{
    if (tweets.statuses.Count > 0)
    {
        foreach (var status in tweets.statuses)
        {
            data.documents.Add(new TwitterDocument(id.ToString(), status.text));
            results.Add(id.ToString(), new Result() { id = id.ToString(), text = status.text,
            id++;
            Console.WriteLine("Id: {0} , tweet: {1}", id, status.text);
        }
    }
}

return JsonConvert.SerializeObject(data);
}
```

If you want to merge the inner if statement with the outer if statement:

1. Place your cursor in the inner if keyword.

2. Press **Ctrl+.** to trigger the **Quick Actions and Refactorings** menu.

```
--  
96  
97     if (tweets.statuses != null)  
98     {  
99         if (tweets.statuses.Count > 0)  
100    Merge with outer 'if' statement >  
101    Invert if  
102    if (tweets.statuses != null)  
103    if (tweets.statuses != null && tweets.statuses.Count > 0)  
104    {  
105        if (tweets.statuses.Count > 0)  
106        {  
107            foreach (var status in tweets.statuses)  
108            {  
109                data.documents.Add(new TwitterDocument(id.ToString(), status.text));  
110                results.Add(id.ToString(), new Result() { id = id.ToString(), text = status.text, userName = status.user.name });  
111                id++;  
112                Console.WriteLine("Id: {0} , tweet: {1}", id, status.text);  
113            }  
114        }  
115    }  
116    return Json...  
117 ...  
118  
119 Preview changes
```

3. Select **Merge with outer if statement**.

```
if (tweets.statuses != null && tweets.statuses.Count > 0)  
{  
    foreach (var status in tweets.statuses)  
    {  
        data.documents.Add(new TwitterDocument(id.ToString(), status.text));  
        results.Add(id.ToString(), new Result() { id = id.ToString(), text = status.text });  
        id++;  
        Console.WriteLine("Id: {0} , tweet: {1}", id, status.text);  
    }  
}  
  
return JsonConvert.SerializeObject(data);  
}
```

See also

- [Refactoring](#)

Sync namespace and folder name

6/17/2019 • 2 minutes to read • [Edit Online](#)

This refactoring applies to:

- C#

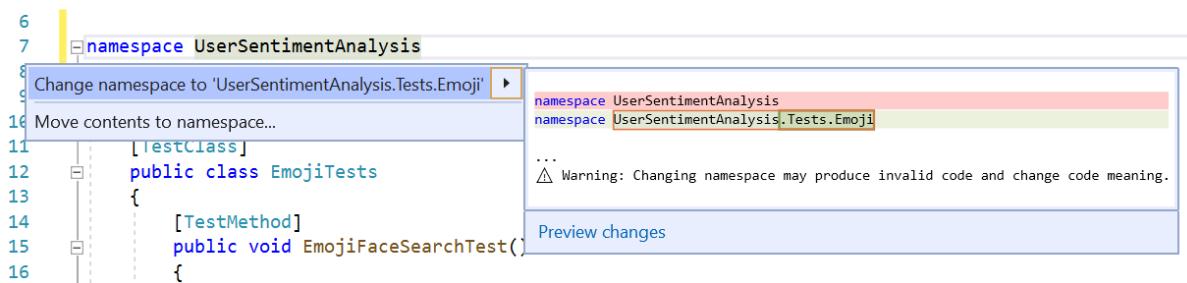
What: Sync namespace and folder name.

When: You want to rearchitect parts of your solution by dragging a file to a new folder.

Why: You want to make sure your namespace keeps up-to date with your new folder structure.

How-to

1. Place your cursor in the namespace name.
2. Press **Ctrl+.** to trigger the **Quick Actions and Refactorings** menu.
3. Select **Change namespace to <folder name>**.



See also

- [Refactoring](#)

Sync a type to a filename, or a filename to a type refactoring

10/18/2019 • 2 minutes to read • [Edit Online](#)

This refactoring applies to:

- C#
- Visual Basic

What: Lets you rename a type to match the filename, or rename a filename to match the type it contains.

When: You have renamed a file or type and haven't yet updated the corresponding file or type to match.

Why: Placing a type in a file with a different name, or vice-versa, it difficult to find what you're looking for. By renaming either the type or filename, code becomes more readable and easier to navigate.

NOTE

This refactoring is not yet available for .NET Standard and .NET Core projects.

How-to

1. Highlight or place the text cursor inside the name of the type to synchronize:

- C#:



- Visual Basic:



2. Next, do one of the following:

- **Keyboard**

- Press **Ctrl+.** to trigger the **Quick Actions and Refactorings** menu and select **Rename file to TypeName.cs** from the Preview window popup, where *TypeName* is the name of the type you have selected.
- Press **Ctrl+.** to trigger the **Quick Actions and Refactorings** menu and select **Rename type to Filename** from the Preview window popup, where *Filename* is the name of the current file.

- **Mouse**

- Right-click the code, select the **Quick Actions and Refactorings** menu, and select **Rename file**

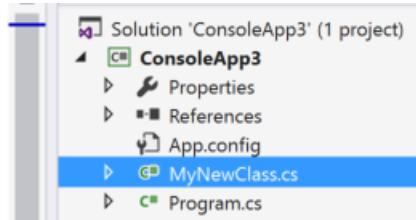
to *TypeName.cs* from the Preview window popup, where *TypeName* is the name of the type you have selected.

- Right-click the code, select the **Quick Actions and Refactorings** menu, and select **Rename type to *Filename*** from the Preview window popup, where *Filename* is the name of the current file.

The type or file is renamed.

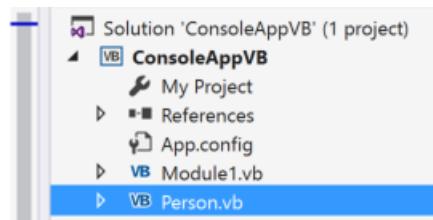
- C#: In the example below, the file **MyClass.cs** was renamed to **MyNewClass.cs** to match the type name.

```
public class MyNewClass
{
    public void Method() { }
}
```



- Visual Basic: In the example below, the file **Employee.vb** was renamed to **Person.vb** to match the type name.

```
Class Person
    Public Sub Method()
        End Sub
    End Class
```



See also

- [Refactoring](#)

Refactoring to replace var with an explicit type

10/18/2019 • 2 minutes to read • [Edit Online](#)

Use this refactoring to replace `var` in a local variable declaration with an explicit type.

This refactoring applies to:

- C#

Why to use an explicit type

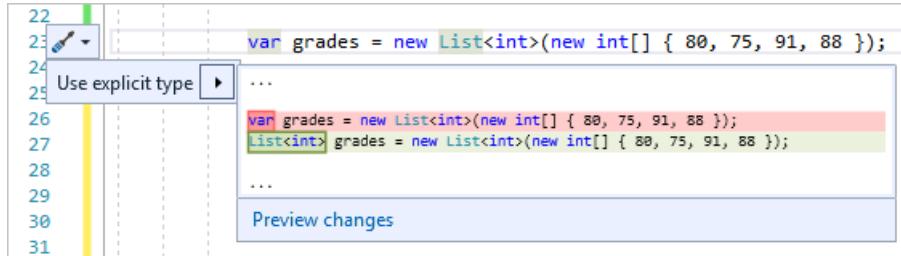
Following are some reasons to declare a variable with an explicit type:

- To improve the code's readability.
- When you don't want to initialize the variable in the declaration.

However, `var` must be used when a variable is initialized with an anonymous type and the properties of the object are accessed at a later point. For more information, see [Implicitly typed local variables \(C#\)](#).

How to use it

1. Place the caret on the `var` keyword.
2. Press **Ctrl+.** or click the screwdriver  icon in the margin of the code file.



3. Select **Use explicit type**. Or, select **Preview changes** to open the **Preview Changes** dialog, and then select **Apply**.

See also

- [Implicitly typed variables \(C#\)](#)
- [Refactoring](#)
- [Preview Changes](#)

Use expression body or block body for lambda expressions

5/10/2019 • 2 minutes to read • [Edit Online](#)

This refactoring applies to:

- C#

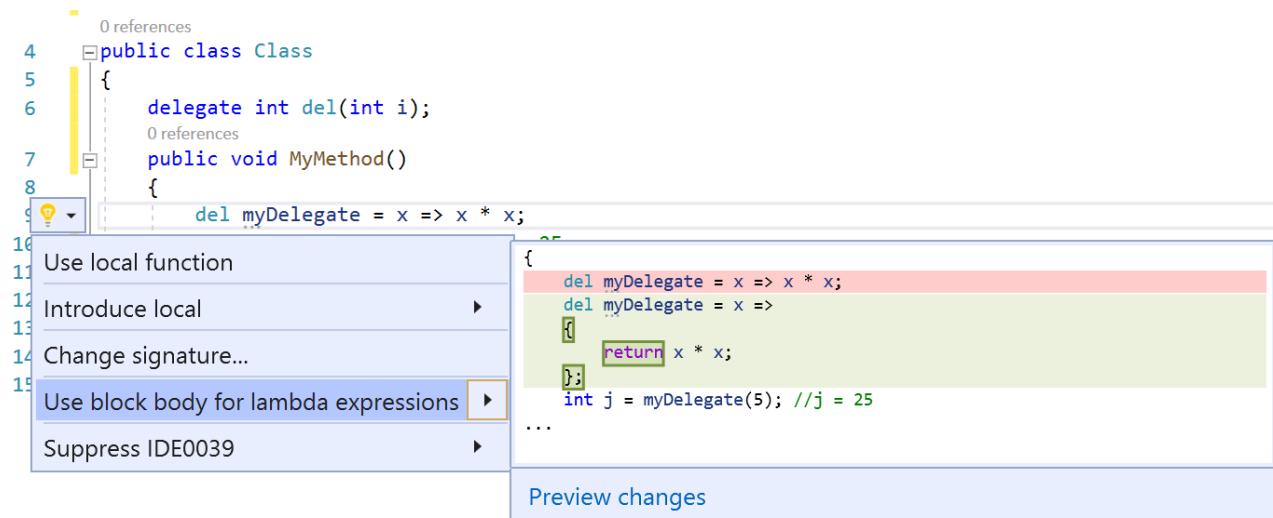
What: Lets you refactor a lambda expression to use an expression body or a block body.

When: You prefer lambda expressions to use either an expression body or a block body.

Why: Lambda expressions can be refactored to improve readability according to your user preference.

Lambda expression body or block body refactoring

1. Place your cursor on the right of a lambda operator.
2. Press **Ctrl+.** to trigger the **Quick Actions and Refactorings** menu.



3. Select **Use block body for lambda expressions** or **Use expression body for lambda expressions**.

See also

- [Refactoring](#)
- [Tips for .NET Developers](#)

Unused value assignments, variables, and parameters

5/10/2019 • 2 minutes to read • [Edit Online](#)

This refactoring applies to:

- C#
- Visual Basic

What: Fades out unused parameters and generates a warning for unused expression values. The compiler also performs a flow analysis to find any unused value assignments. Unused value assignments fade out and a light bulb appears with a [Quick Action](#) to remove the redundant assignment. Unused variables with unknown values show a [Quick Action](#) suggestion to use [discards](#) instead. (Discards are temporary, dummy variables that are intentionally unused in application code. They can reduce memory allocation and make your code easier to read.)

When: You have value assignments, parameters, or expression values that are never used.

Why: Sometimes it's difficult to tell if a value assignment, variable, or parameter is no longer being used. By fading out these values or generating a warning, you get a visual cue of what code you can delete.

Unused expression values and parameters diagnostic

1. Have any value assignment, variable, or parameter that isn't used.
2. The unused value assignment or parameter appears faded out. The unused expression value generates a warning.

```
0 references
public class Class
{
    0 references
    public void MyMethod(int a, int b, string c){}
}
```

[💡] (parameter) int a

Remove unused parameter 'a' if it is not part of a shipped public API

Show potential fixes (Alt+Enter or Ctrl+.)


```
3     0 references
4     class Class
5     {
6         0 references
7         public void MyMethod()
8         {
9             string notUsed = "Hi";
10        }
11 }
```

💡 Remove unused variable ►

⚠️ CS0219 The variable 'notUsed' is assigned but its value is never used

Suppress CS0219 ►

Preview changes

Fix all occurrences in: Document | Project | Solution

The screenshot shows a code editor with two windows open. The top window displays the following C# code:

```
41 1 reference
42 int M()
43 {
44     int x = 1;
45 }
46 Remove redundant assignment ▾
47 Inline temporary variable
48 Suppress IDE0059 ▾
49
50
51
52
53
54 Fix all occurrences in: Document | Project | Solution
55
```

The bottom window displays the following C# code:

```
63 0 references
64 void M()
65 {
66     int x = M2();
67 }
68 Use discard '_' ▾
69 Fix formatting
70 Suppress IDE0059 ▾
71
72
73
74
75 --
```

Both windows show a tooltip for the IDE0059 rule: "Value assigned to 'x' is never used". The tooltip includes a preview of the refactored code and a link to fix all occurrences.

See also

- [Refactoring](#)
- [Tips for .NET Developers](#)

Wrap and align call chains

8/19/2019 • 2 minutes to read • [Edit Online](#)

This refactoring applies to:

- C#

What: Lets you wrap and align chains of method calls.

When: You have a long chain consisting of several method calls in one statement.

Why: Reading a long list is easier when they're wrapped or indented according to user preference.

How-to

1. Place your cursor in any of the call chains.
2. Press **Ctrl+.** to trigger the **Quick Actions and Refactorings** menu.
3. Select **Wrap call chain** or **Wrap and align call chain** to accept the refactoring.



See also

- [Refactoring](#)

Wrap, indent, and align parameters or arguments

5/22/2019 • 2 minutes to read • [Edit Online](#)

This refactoring applies to:

- C#
- Visual Basic

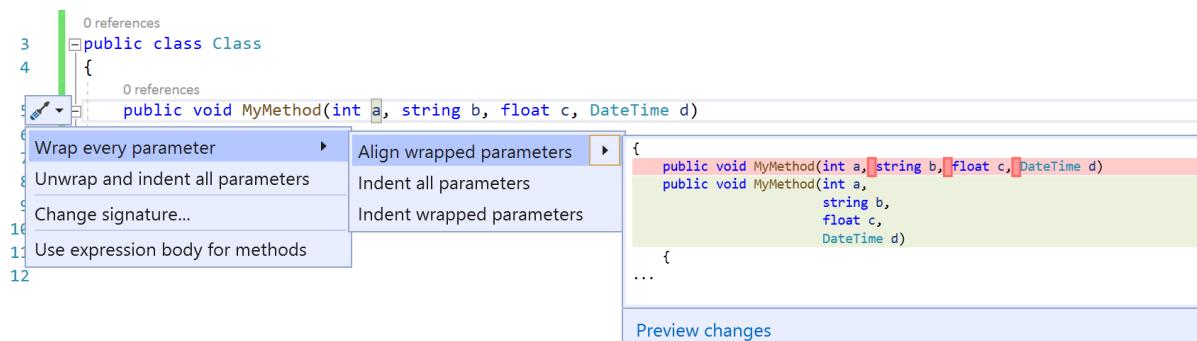
What: Lets you wrap, indent, and align parameters or arguments.

When: You have a method declaration or call that has multiple parameters or arguments.

Why: Reading a long list of parameters or arguments is easier when they're wrapped or indented according to user preference.

How-to

1. Place your cursor in a parameter list.
2. Press **Ctrl+.** to trigger the **Quick Actions and Refactorings** menu.



3. Press **Enter** to accept the refactoring.

```
0 references
public void MyMethod(int a,
                     string b,
                     float c,
                     DateTime d)
{
    Console.WriteLine(a.ToString(), b, c, d);
}
```

See also

- [Refactoring](#)

Walkthrough: Test-first development with the Generate From Usage feature

10/18/2019 • 7 minutes to read • [Edit Online](#)

This topic demonstrates how to use the [Generate From Usage](#) feature, which supports test-first development.

Test-first development is an approach to software design in which you first write unit tests based on product specifications, and then write the source code that is required to make the tests succeed. Visual Studio supports test-first development by generating new types and members in the source code when you first reference them in your test cases, before they are defined.

Visual Studio generates the new types and members with minimal interruption to your workflow. You can create stubs for types, methods, properties, fields, or constructors without leaving your current location in code. When you open a dialog box to specify options for type generation, the focus returns immediately to the current open file when the dialog box closes.

The **Generate From Usage** feature can be used with test frameworks that integrate with Visual Studio. In this topic, the Microsoft Unit Testing Framework is demonstrated.

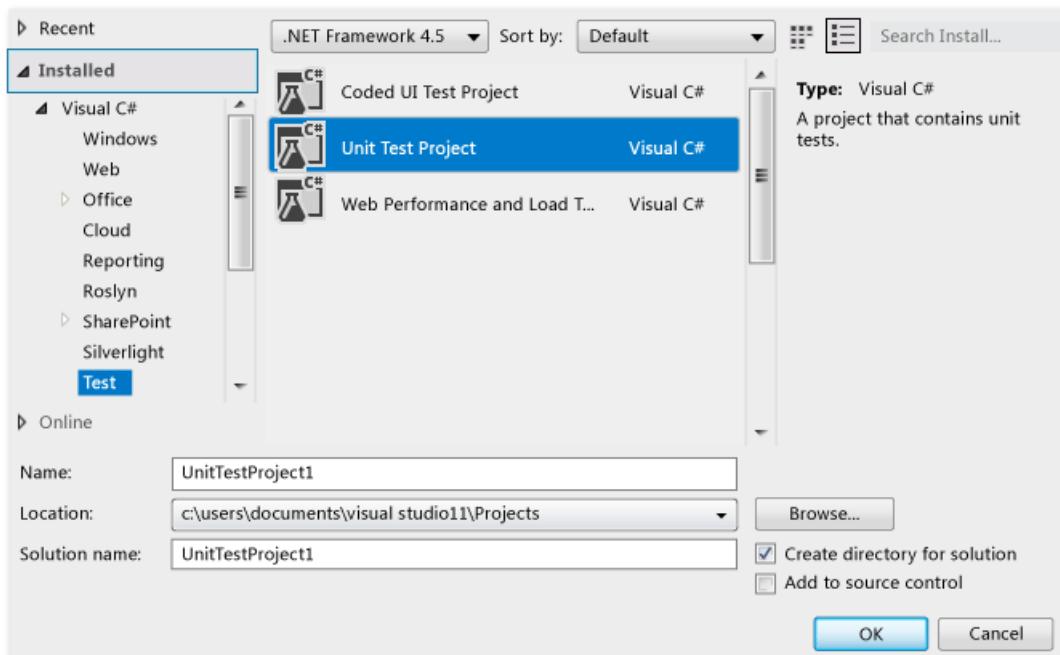
NOTE

Your computer might show different names or locations for some of the Visual Studio user interface elements in this article. You may be using a different edition of Visual Studio or different environment settings. For more information, see [Personalize the IDE](#).

Create a Windows Class Library project and a Test project

1. In C# or Visual Basic, create a new **Windows Class Library** project. Name it `GFUDemo_VB` or `GFUDemo_CS`, depending on which language you are using.
2. In **Solution Explorer**, right-click the solution icon at the top, choose **Add > New Project**.
3. Create a new **Unit Test Project (.NET Framework)** project.

The following illustration shows the **New Project** dialog box for C# templates.



Add a reference to the Class Library project

1. In **Solution Explorer**, under your unit test project, right-click the **References** entry and choose **Add Reference**.
2. In the **Reference Manager** dialog box, select **Projects** and then select the class library project.
3. Choose **OK** to close the **Reference Manager** dialog box.
4. Save your solution. You are now ready to begin writing tests.

Generate a new class from a unit test

1. The test project contains a file that is named *UnitTest1*. Double-click this file in **Solution Explorer** to open it in the code editor. A test class and test method have been generated.
2. Locate the declaration for class `UnitTest1` and rename it to `AutomobileTest`.

NOTE

IntelliSense now provides two alternatives for IntelliSense statement completion: *completion mode* and *suggestion mode*. Use suggestion mode for situations in which classes and members are used before they are defined. When an **IntelliSense** window is open, you can press **Ctrl+Alt+Space** to toggle between completion mode and suggestion mode. See [Use IntelliSense](#) for more information. Suggestion mode will help when you are typing `Automobile` in the next step.

3. Locate the `TestMethod1()` method and rename it to `DefaultAutomobileIsInitializedCorrectly()`. Inside this method, create a new instance of a class named `Automobile`, as shown in the following screenshots. A wavy underline appears, which indicates a compile-time error, and a **Quick Actions** error light bulb appears in the left margin, or directly below the squiggle if you hover over it.

```

6  <TestMethod()> Public Sub DefaultAutomobileIsInitializedCorrectly()
7    Dim myAuto = New Automobile()
8  End Sub
9
10 End Class
11

```

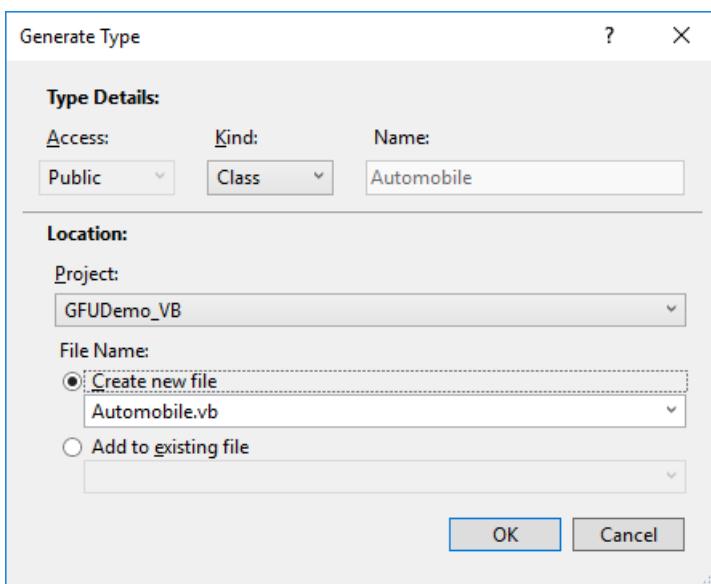
The code editor shows a wavy underline under the word `Automobile` in the line `Dim myAuto = New Automobile()`. A tooltip box appears with the message "Type 'Automobile' is not defined." and a link "Show potential fixes (Alt+Enter or Ctrl+.)".

```

9 [TestMethod]
10 0 references | 0 changes | 0 authors, 0 changes
11 public void DefaultAutomobileIsInitializedCorrectly()
12 {
13     Automobile myAuto = new Automobile();

```

4. Choose or click the **Quick Actions** light bulb. You'll see an error message that states that the type `Automobile` is not defined. You are also presented with some solutions.
5. Click **Generate new type** to open the **Generate Type** dialog box. This dialog box provides options that include generating the type in a different project.
6. In the **Project** list, click **GFUDemo_VB** or **GFUDemo_CS** to instruct Visual Studio to add the file to the class library project instead of the test project. If it's not already selected, choose **Create new file** and name it *Automobile.cs* or *Automobile.vb*.



7. Click **OK** to close the dialog box and create the new file.
8. In **Solution Explorer**, look under the **GFUDemo_VB** or **GFUDemo_CS** project node to verify that the new *Automobile.vb* or *Automobile.cs* file is there. In the code editor, the focus is still in `DefaultAutomobileIsInitializedCorrectly`, which enables you to continue to write your test with a minimum of interruption.

Generate a property stub

Assume that the product specification states that the `Automobile` class has two public properties named `Model` and `TopSpeed`. These properties must be initialized with default values of "Not specified" and -1 by the default constructor. The following unit test will verify that the default constructor sets the properties to their correct default values.

1. Add the following line of code to the `DefaultAutomobileIsInitializedCorrectly` test method.

```
Assert.IsTrue((myAuto.Model == "Not specified") && (myAuto.TopSpeed == -1));
```

```
Assert.IsTrue((myAuto.Model = "Not specified") And (myAuto.TopSpeed = -1))
```

2. Because the code references two undefined properties on `Automobile`, a wavy underline appears under `Model` and `TopSpeed`. Hover over `Model` and choose the **Quick Actions** error light bulb, then choose **Generate property 'Automobile.Model'**.

3. Generate a property stub for the `TopSpeed` property in the same way.

In the `Automobile` class, the types of the new properties are correctly inferred from the context.

Generate a stub for a new constructor

Now we'll create a test method that will generate a constructor stub to initialize the `Model` and `TopSpeed` properties. Later, you'll add more code to complete the test.

1. Add the following additional test method to your `AutomobileTest` class.

```
[TestMethod]
public void AutomobileWithModelNameCanStart()
{
    string model = "Contoso";
    int topSpeed = 199;
    Automobile myAuto = new Automobile(model, topSpeed);
}
```

```
<TestMethod()> Public Sub AutomobileWithModelNameCanStart()
    Dim model As String = "Contoso"
    Dim topSpeed As Integer = 199
    Dim myAuto As New Automobile(model, topSpeed)
End Sub
```

2. Click the **Quick Actions** error light bulb under the red squiggle, and then click **Generate constructor in 'Automobile'**.

In the `Automobile` class file, notice that the new constructor has examined the names of the local variables that are used in the constructor call, found properties that have the same names in the `Automobile` class, and supplied code in the constructor body to store the argument values in the `Model` and `TopSpeed` properties.

3. After you generate the new constructor, a wavy underline appears under the call to the default constructor in `DefaultAutomobileIsInitializedCorrectly`. The error message states that the `Automobile` class has no constructor that takes zero arguments. To generate an explicit default constructor that does not have parameters, click the **Quick Actions** error light bulb, and then click **Generate constructor in 'Automobile'**.

Generate a stub for a method

Assume that the specification states that a new `Automobile` can be put into a `IsRunning` state if its `Model` and `TopSpeed` properties are set to something other than the default values.

1. Add the following lines to the `AutomobileWithModelNameCanStart` method.

```
myAuto.Start();
Assert.IsTrue(myAuto.IsRunning == true);
```

```
myAuto.Start()
Assert.IsTrue(myAuto.IsRunning = True)
```

2. Click the **Quick Actions** error light bulb for the `myAuto.Start` method call and then click **Generate method 'Automobile.Start'**.
3. Click the **Quick Actions** light bulb for the `IsRunning` property and then click **Generate property 'Automobile.IsRunning'**.

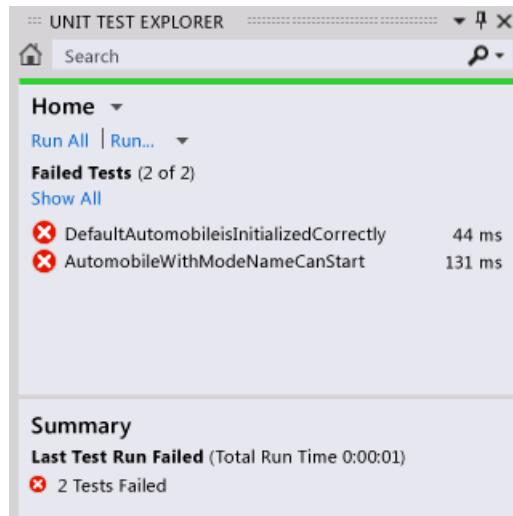
The `Automobile` class now contains a method named `Start()` and a property named `IsRunning`.

Run the tests

1. On the **Test** menu, choose **Run > All Tests**.

The **Run > All Tests** command runs all the tests in any test frameworks that are written for the current solution. In this case, there are two tests, and they both fail, as expected. The `DefaultAutomobileIsInitializedCorrectly` test fails because the `Assert.IsTrue` condition returns `False`. The `AutomobileWithModelNameCanStart` test fails because the `Start` method in the `Automobile` class throws an exception.

The **Test Results** window is shown in the following illustration.



2. In the **Test Results** window, double-click on each test result row to go to the location of each test.

Implement the source code

1. Add the following code to the default constructor so that the `Model`, `TopSpeed` and `IsRunning` properties are all initialized to their correct default values of "Not specified", -1, and `False` (or `false` for C#).

```
public Automobile()
{
    this.Model = "Not specified";
    this.TopSpeed = -1;
    this.IsRunning = false;
}
```

```
Sub New()
    Model = "Not specified"
    TopSpeed = -1
    IsRunning = False
End Sub
```

2. When the `Start` method is called, it should set the `IsRunning` flag to true only if the `Model` or `TopSpeed` properties are set to something other than their default value. Remove the `NotImplementedException` from the method body and add the following code.

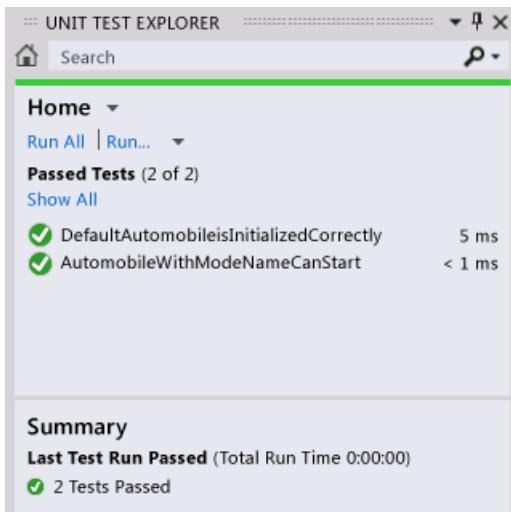
```
public void Start()
{
    if (this.Model != "Not specified" || this.TopSpeed != -1)
        this.IsRunning = true;
    else
        this.IsRunning = false;
}
```

```
Sub Start()
    If Model <> "Not specified" Or TopSpeed <> -1 Then
        IsRunning = True
    Else
        IsRunning = False
    End If
End Sub
```

Run the tests again

- On the **Test** menu, point to **Run**, and then click **All Tests**.

This time the tests pass. The **Test Results** window is shown in the following illustration.



See also

- [Generate from usage](#)
- [Features of the code editor](#)
- [Use IntelliSense](#)
- [Unit test your code](#)
- [Quick Actions](#)

IntelliSense in Visual Studio

10/18/2019 • 5 minutes to read • [Edit Online](#)

IntelliSense is a code-completion aid that includes a number of features: List Members, Parameter Info, Quick Info, and Complete Word. These features help you to learn more about the code you're using, keep track of the parameters you're typing, and add calls to properties and methods with only a few keystrokes.

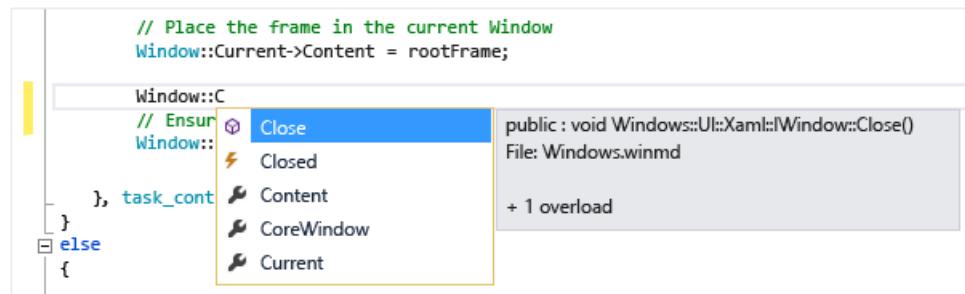
Many aspects of IntelliSense are language-specific. For more information about IntelliSense for different languages, see the topics listed in the [See also](#) section.

List Members

A list of valid members from a type (or namespace) appears after you type a trigger character (for example, a period (.) in managed code or :: in C++). If you continue typing characters, the list is filtered to include only the members that begin with those characters or where the beginning of *any* word within the name starts with those characters. IntelliSense also performs "camel case" matching, so you can just type the first letter of each camel-cased word in the member name to see the matches.

After selecting an item, you can insert it into your code by pressing **Tab** or by typing a space. If you select an item and type a period, the item appears followed by the period, which brings up another member list. When you select an item but before you insert it, you get Quick Info for the item.

In the member list, the icon to the left represents the type of the member, such as namespace, class, function, or variable. For a list of icons, see [Class View and Object Browser icons](#). The list may be quite long, so you can press **PgUp** and **PgDn** to move up or down in the list.



You can invoke the **List Members** feature manually by typing **Ctrl+J**, choosing **Edit > IntelliSense > List Members**, or by choosing the **List Members** button on the editor toolbar. When it is invoked on a blank line or outside a recognizable scope, the list displays symbols in the global namespace.

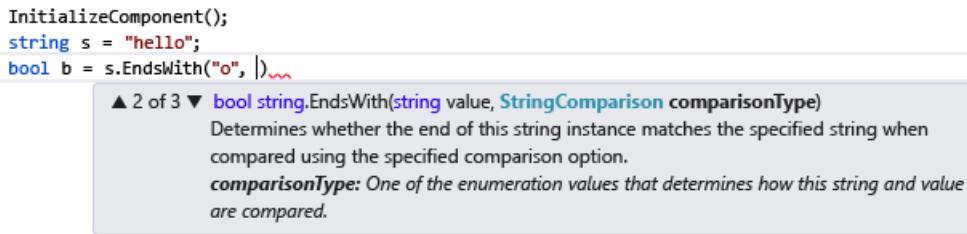
To turn List Members off by default (so that it does not appear unless specifically invoked), go to **Tools > Options > All Languages** and deselect **Auto list members**. If you want to turn off List Members only for a specific language, go to the **General** settings for that language.

You can also change to suggestion mode, in which only the text you type is inserted into the code. For example, if you enter an identifier that is not in the list and press **Tab**, in completion mode the entry would replace the typed identifier. To toggle between completion mode and suggestion mode, press **Ctrl+Alt+Space**, or choose **Edit > IntelliSense > Toggle Completion Mode**.

Parameter Info

Parameter Info gives you information about the number, names, and types of parameters required by a method, attribute generic type parameter (in C#), or template (in C++).

The parameter in bold indicates the next parameter that is required as you type the function. For overloaded functions, you can use the **Up** and **Down** arrow keys to view alternative parameter information for the function overloads.

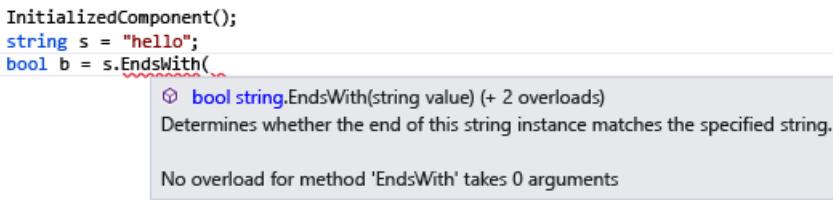


When you annotate functions and parameters with XML Documentation comments, the comments will display as Parameter Info. For more information, see [Supply XML code comments](#).

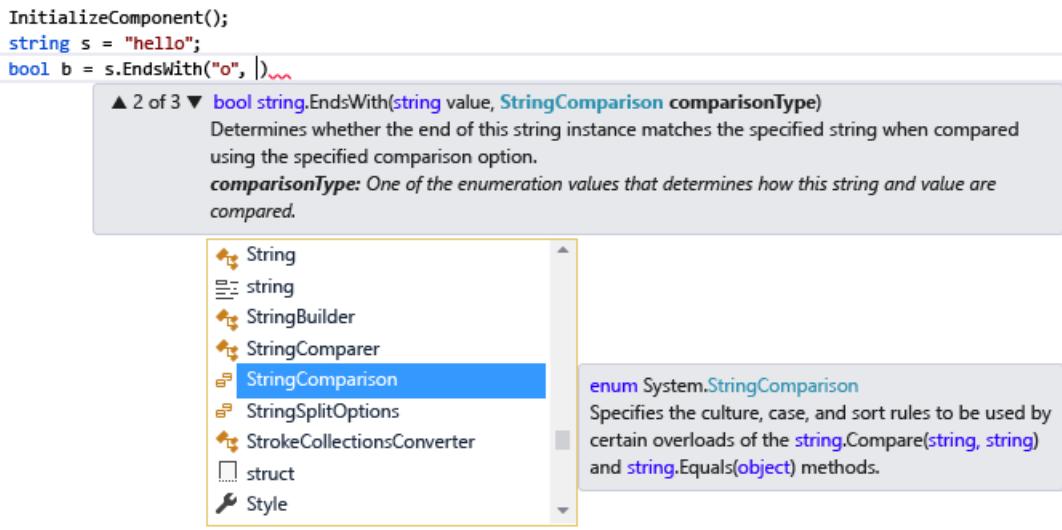
You can manually invoke Parameter Info by choosing **Edit > IntelliSense > Parameter Info**, by pressing **Ctrl+Shift+Space**, or by choosing the **Parameter Info** button on the editor toolbar.

Quick Info

Quick Info displays the complete declaration for any identifier in your code.



When you select a member from the **List Members** box, Quick Info also appears.



You can manually invoke Quick Info by choosing **Edit > IntelliSense > Quick Info**, by pressing **Ctrl+I**, or by choosing the **Quick Info** button on the editor toolbar.

If a function is overloaded, IntelliSense may not display information for all forms of the overload.

You can turn Quick Info off for C++ code by navigating to **Tools > Options > Text Editor > C/C++ > Advanced**, and setting **Auto Quick Info** to `false`.

Complete Word

Complete Word completes the rest of a variable, command, or function name after you have entered enough

characters to disambiguate the term. You can invoke Complete Word by choosing **Edit > IntelliSense > Complete Word**, by pressing **Ctrl+Space**, or by choosing the **Complete Word** button on the editor toolbar.

IntelliSense options

IntelliSense options are on by default. To turn them off, choose **Tools > Options > Text Editor** and deselect **Parameter information** or **Auto list members** if you do not want the List Members feature.

IntelliSense icons

The icons in IntelliSense can convey additional meaning with icon modifiers. These are stars, hearts, and locks layered on top of the object's icon that convey protected, internal, or private, respectively.

ICON	ACCESSIBILITY	DESCRIPTION
	Public class	Access is not restricted.
	Protected class	Access is limited to the containing class or types derived from the containing class.
	Protected internal class	Access is limited to the current assembly or types derived from the containing class.
	Internal class	Access is limited to the current assembly.
	Private class	Access is limited to the containing class or types derived from the containing class within the current assembly. (Available since C# 7.2.)

Troubleshoot IntelliSense

The IntelliSense options may not work as you expect in certain cases.

The cursor is below a code error. You might not be able to use IntelliSense if an incomplete function or other error exists in the code above the cursor because IntelliSense might not be able to parse the code elements. You can resolve this problem by commenting out the applicable code.

The cursor is in a code comment. You can't use IntelliSense if the cursor is in a comment in your source file.

The cursor is in a string literal. You can't use IntelliSense if the cursor is in the quotation marks around a string literal, as in the following example:

```
MessageBox( hWnd, "String literal|")
```

The automatic options are turned off. By default, IntelliSense works automatically, but you can disable it. Even if automatic statement completion is disabled, you can invoke an IntelliSense feature.

See also

- [Visual Basic IntelliSense](#)

- C# IntelliSense
- JavaScript IntelliSense
- Write and refactor code (C++)
- Supply XML code comments

IntelliSense for Visual Basic code files

10/18/2019 • 2 minutes to read • [Edit Online](#)

The Visual Basic source code editor offers the following IntelliSense features:

Syntax tips

Syntax tips display the syntax of the statement that you are typing. This is useful for statements such as [Declare](#).

Automatic completion

- Completion on various keywords

For example, if you type `goto` and a space, IntelliSense displays a list of the defined labels in a drop-down menu. Other supported keywords include `Exit`, `Implements`, `Option`, and `Declare`.

- Completion on `Enum` and `Boolean`

When a statement will refer to a member of an enumeration, IntelliSense displays a list of the members of the `Enum`. When a statement will refer to a `Boolean`, IntelliSense displays a true-false drop-down menu.

Completion can be turned off by default by deselecting **Auto list members** from the **General** property page in the **Visual Basic** folder.

You can manually invoke completion by invoking List Members, Complete Word, or **Alt+Right Arrow**. For more information, see [Use IntelliSense](#).

IntelliSense in Zone

IntelliSense in Zone assists Visual Basic developers who need to deploy applications through ClickOnce and are constrained to partial trust settings. This feature:

- Enables you to choose the permissions the application will run with.
- Display APIs in the chosen Zone as available in List Members, and display APIs that require additional permissions as unavailable.

For more information, see [Code access security for ClickOnce applications](#).

Filtered completion lists

In Visual Basic, IntelliSense completion lists have two tab controls located near the bottom of the lists. The **Common** tab, which is selected by default, displays items that are most often used to complete the statement that you are writing. The **All** tab displays all items that are available for automatic completion, including those that are also on the **Common** tab.

See also

- [Use IntelliSense](#)

C# IntelliSense

10/18/2019 • 8 minutes to read • [Edit Online](#)

C# IntelliSense is available when coding in the editor, and while debugging in the [Immediate mode](#) command window.

Completion lists

The IntelliSense completion lists in C# contain tokens from List Members, Complete Word, and more. It provides quick access to:

- Members of a type or namespace
- Variables, commands, and functions names
- Code snippets
- Language keywords
- Extension methods

The completion list in C# is also smart enough to filter out irrelevant tokens and pre-select a token based on context. For more information, see [Filtered completion lists](#).

Code snippets in completion lists

In C#, the completion list includes code snippets to help you easily insert predefined bodies of code into your program. Code snippets appear in the completion list as the snippet's [shortcut text](#). For more information about code snippets that are available in C# by default, see [C# code snippets](#).

Language keywords in completion lists

In C#, the completion list also includes language keywords. For more information about C# language keywords, see [C# keywords](#).

Extension methods in completion lists

In C#, the completion list includes extension methods that are in scope.

NOTE

The completion list does not display all extension methods for [String](#) objects.

Extension methods use a different icon than instance methods. For a list icon reference guide, see [Class View and Object Browser icons](#). When an instance method and extension method with the same name are both in scope, the completion list displays the extension method icon.

Filtered completion lists

IntelliSense removes unnecessary members from the completion list by using filters. C# filters the completion lists that appear for these items:

- **Interfaces and base classes:** IntelliSense automatically removes items from the interface and base class completion lists, in both class declaration base and interface lists and constraint lists. For example, enums do not appear in the completion list for base classes, because enums cannot be used for base classes. The completion list of base classes only contains interfaces and namespaces. If you select an item in the list and

then type a comma, IntelliSense removes base classes from the completion list because C# does not support multiple inheritance. The same behavior occurs for constraint clauses also.

- **Attributes:** When you apply an attribute to a type, the completion list is filtered so that the list only contains those types that descend from the namespaces that contain those types, such as [Attribute](#).
- **Catch clauses**
- **Object initializers:** Only members that can be initialized will appear in the completion list.
- **new keyword:** When you type `new` and then press the **Space**, a completion list appears. An item is automatically selected in the list, based on the context in your code. For example, items are automatically selected in the completion list for declarations and for return statements in methods.
- **enum keyword:** When you press the **Space** after an equal sign for an enum assignment, a completion list appears. An item is automatically selected in the list, based on the context in your code. For example, items are automatically selected in the completion list after you type the keyword `return` and when you make a declaration.
- **as and is operators:** A filtered completion list is displayed automatically when you press the **Space** after you have typed the `as` or `is` keyword.
- **Events:** When you type the keyword `event`, the completion list only contains delegate types.
- **Parameter help** automatically sorts to the first method overload that matches the parameters as you enter them. If multiple method overloads are available, you can use the up and down arrows to navigate to the next possible overload in the list.

Most recently used members

IntelliSense remembers the members that you have recently selected in the pop-up [List Members](#) box for automatic object name completion. The next time you use **Member List**, the most recently used members are shown at the top. The history of most recently used members is cleared between each Visual Studio session.

override

When you type `override` and then press **Space**, IntelliSense displays all of the valid base class members that you can override in a pop-up list box. Typing the return type of the method after `override` prompts IntelliSense to only show methods that return the same type. When IntelliSense cannot find any matches, it displays all of the base class members.

AI-enhanced IntelliSense

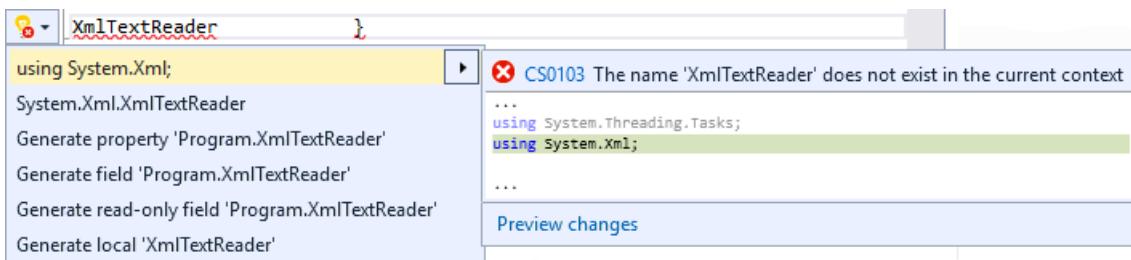
[Visual Studio IntelliCode](#) provides artificial intelligence-enhanced IntelliSense completion lists. IntelliCode predicts the most likely correct API to use rather than just presenting an alphabetical list of members. It uses your current code context and patterns to provide the dynamic list.

Automatic code generation

Add using

The **Add using** IntelliSense operation automatically adds the required `using` directive to your code file. This feature enables you to maintain your focus on the code you are writing rather than requiring you to shift your focus to another part of the code.

To initiate the **Add using** operation, position the cursor on a type reference that cannot be resolved. For example, when you create a console application and then add `XmlReader` to the body of the `Main` method, a red squiggle appears on that line of code because the type reference cannot be resolved. You can then invoke the **Add using** through the **Quick Actions**. The **Quick Actions** are only visible when the cursor is positioned on the unbound type.



Click the error light bulb icon, and then choose **using System.Xml;** to automatically add the using directive.

Remove and sort usings

The **Remove and Sort Usings** option sorts and removes `using` and `extern` declarations without changing the behavior of the source code. Over time, source files may become bloated and difficult to read because of unnecessary and unorganized `using` directives. The **Remove and Sort Usings** option compacts source code by removing unused `using` directives, and improves readability by sorting them. On the **Edit** menu, choose **IntelliSense**, and then choose **Organize Usings**.

Implement interface

IntelliSense provides an option to help you implement an **interface** while working in the code editor. Normally, to implement an interface properly, you must create a method declaration for every member of the interface in your class. Using IntelliSense, after you type the name of an interface in a class declaration, a **Quick Actions** light bulb is displayed. The light bulb gives you the option to implement the interface automatically, using explicit or implicit naming. Under explicit naming, the method declarations carry the name of the interface. Under implicit naming, the method declarations do not indicate the interface to which they belong. An explicitly named interface method can only be accessed through an interface instance, and not through a class instance. For more information, see [Explicit interface implementation](#).

Implement Interface generates the minimum number of method stubs that's required to satisfy the interface. If a base class implements parts of the interface, then those stubs aren't regenerated.

Implement abstract base class

IntelliSense provides an option to help you implement members of an abstract base class automatically while working in the code editor. Normally, to implement members of an abstract base class requires creating a new method definition for each method of the abstract base class in your derived class. Using IntelliSense, after typing the name of an abstract base class in a class declaration, a **Quick Actions** light bulb is displayed. The light bulb gives you the option to implement the base class methods automatically.

The method stubs that are generated by the **Implement Abstract Base Class** feature are modeled by the code snippet defined in the file *MethodStub.snippet*. Code snippets are modifiable. For more information, see [Walkthrough: Create a code snippet](#).

Generate from usage

The **Generate From Usage** feature enables you to use classes and members before you define them. You can generate a stub for any class, constructor, method, property, field, or enum that you want to use but have not yet defined. You can generate new types and members without leaving your current location in code. This minimizes interruption to your workflow.

A red wavy underline appears under each undefined identifier. When you rest the mouse pointer on the identifier, an error message appears in a tooltip. To display the appropriate options, you can use one of the following procedures:

- Click the undefined identifier. A **Quick Actions** error light bulb appears under the identifier. Click the error light bulb.
- Click the undefined identifier, and then press **Ctrl+.** (**Ctrl** + period).

- Right-click the undefined identifier, and then click **Quick Actions and Refactorings**.

The options that appear can include the following:

- **Generate property**
- **Generate field**
- **Generate method**
- **Generate class**
- **Generate new type** (for a class, struct, interface, or enum)

Generate event handlers

In the code editor, IntelliSense can help you hook up methods (event handlers) to event fields.

When you type the `+=` operator after an event field in a .cs file, IntelliSense prompts you with the option to press the **Tab** key. This inserts a new instance of a delegate that points to the method handling the event.

```
button1.Click +=  
    new EventHandler(button1_Click); (Press TAB to insert)
```

If you press **Tab**, IntelliSense automatically finishes the statement for you and displays the event handler reference as selected text in the code editor. To complete the automatic event hookup, IntelliSense prompts you to press the **Tab** key again to create an empty stub for the event handler.

```
button1.Click +=new EventHandler(button1_Click);  
    Press TAB to generate handler 'button1_Click' in this class
```

NOTE

If a new delegate that is created by IntelliSense references an existing event handler, IntelliSense communicates this information in the tooltip. You can then modify this reference; the text is already selected in the code editor. Otherwise, automatic event hookup is complete at this point.

If you press **Tab**, IntelliSense stubs out a method with the correct signature and puts the cursor in the body of your event handler.

NOTE

Use the **Navigate Backward** command on the **View** menu (**Ctrl+-**) to go back to the event hookup statement.

See also

- [Use IntelliSense](#)
- [Visual Studio IDE](#)

JavaScript IntelliSense

3/21/2019 • 4 minutes to read • [Edit Online](#)

Visual Studio provides a powerful JavaScript editing experience right out of the box. Powered by a TypeScript based language service, Visual Studio delivers richer IntelliSense, support for modern JavaScript features, and improved productivity features such as Go to Definition, refactoring, and more.

NOTE

Starting in Visual Studio 2017, the JavaScript language service uses a new engine for the language service (called "Salsa"). Details are included in this article, and you can also read this [blog post](#). The new editing experience also mostly applies to Visual Studio Code. See the [VS Code docs](#) for more info.

For more information about the general IntelliSense functionality of Visual Studio, see [Using IntelliSense](#).

What's new in the JavaScript language service in Visual Studio 2017

Starting in Visual Studio 2017, JavaScript IntelliSense displays a lot more information on parameter and member lists. This new information is provided by the TypeScript language service, which uses static analysis behind the scenes to better understand your code.

TypeScript uses several sources to build up this information:

- [IntelliSense based on type inference](#)
- [IntelliSense based on JSDoc](#)
- [IntelliSense based on TypeScript declaration files](#)
- [Automatic acquisition of type definitions](#)

IntelliSense based on type inference

In JavaScript, most of the time there is no explicit type information available. Luckily, it is usually fairly easy to figure out a type given the surrounding code context. This process is called type inference.

For a variable or property, the type is typically the type of the value used to initialize it or the most recent value assignment.

```
var nextItem = 10;
nextItem; // here we know nextItem is a number

nextItem = "box";
nextItem; // now we know nextItem is a string
```

For a function, the return type can be inferred from the return statements.

For function parameters, there is currently no inference, but there are ways to work around this using JSDoc or TypeScript *.d.ts* files (see later sections).

Additionally, there is special inference for the following:

- "ES3-style" classes, specified using a constructor function and assignments to the `prototype` property.
- CommonJS-style module patterns, specified as property assignments on the `exports` object, or assignments to the `module.exports` property.

```
function Foo(param1) {
  this.prop = param1;
}
Foo.prototype.getIt = function () { return this.prop; };
// Foo will appear as a class, and instances will have a 'prop' property and a 'getIt' method.

exports.Foo = Foo;
// This file will appear as an external module with a 'Foo' export.
// Note that assigning a value to "module.exports" is also supported.
```

IntelliSense based on JSDoc

Where type inference does not provide the desired type information (or to support documentation), type information may be provided explicitly via JSDoc annotations. For example, to give a partially declared object a specific type, you can use the `@type` tag as shown below:

```
/*
 * @type {{a: boolean, b: boolean, c: number}}
 */
var x = {a: true};
x.b = false;
x. // <- "x" is shown as having properties a, b, and c of the types specified
```

As mentioned, function parameters are never inferred. However, using the JSDoc `@param` tag you can add types to function parameters as well.

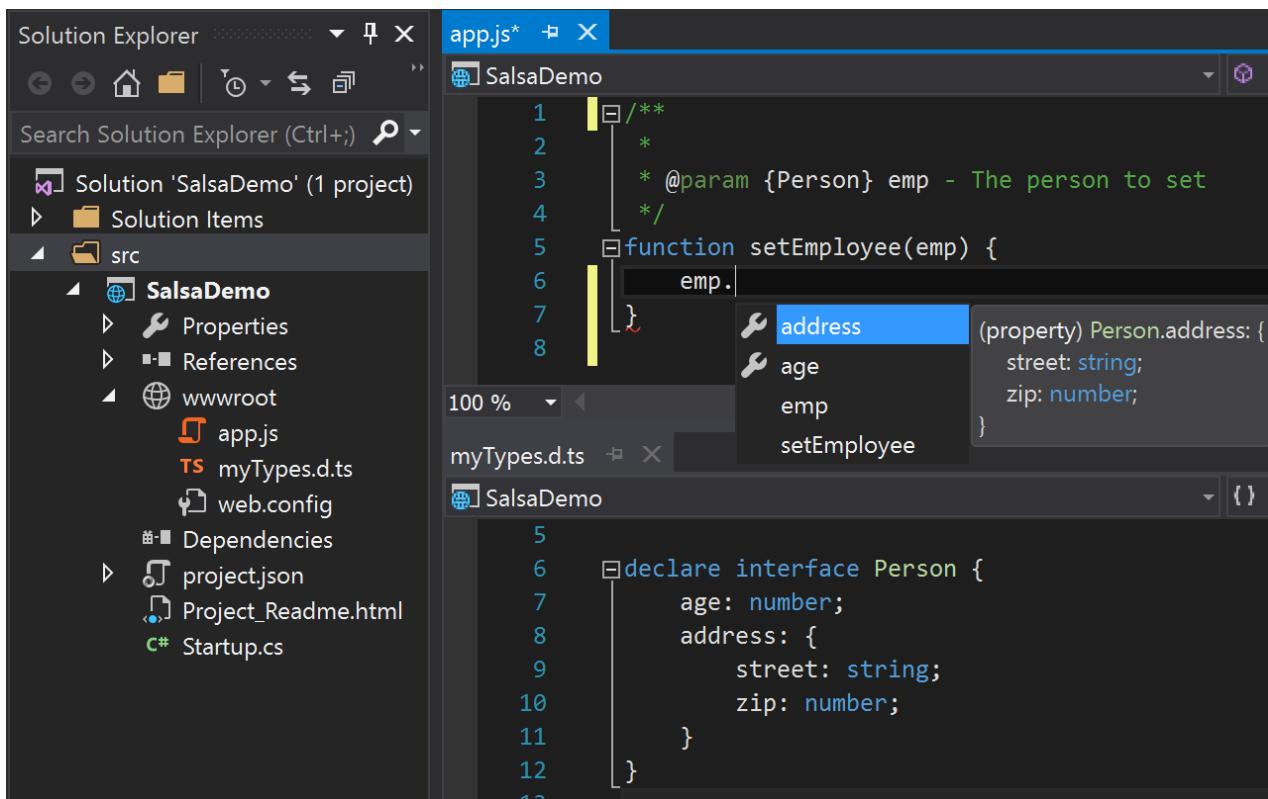
```
/*
 * @param {string} param1 - The first argument to this function
 */
function Foo(param1) {
  this.prop = param1; // "param1" (and thus "this.prop") are now of type "string".
}
```

See [JSDoc support in JavaScript](#) for the JSDoc annotations currently supported.

IntelliSense based on TypeScript declaration files

Because JavaScript and TypeScript are now based on the same language service, they are able to interact in a richer way. For example, JavaScript IntelliSense can be provided for values declared in a `.d.ts` file (see [TypeScript documentation](#)), and types such as interfaces and classes declared in TypeScript are available for use as types in JSDoc comments.

Below, we show a simple example of a TypeScript definition file providing such type information (via an interface) to a JavaScript file in the same project (using a `JsDoc` tag).



Automatic acquisition of type definitions

In the TypeScript world, most popular JavaScript libraries have their APIs described by `.d.ts` files, and the most common repository for such definitions is on [DefinitelyTyped](#).

By default, the Salsa language service will try to detect which JavaScript libraries are in use and automatically download and reference the corresponding `.d.ts` file that describes the library in order to provide richer IntelliSense. The files are downloaded to a cache located under the user folder at `%LOCALAPPDATA%\Microsoft\TypeScript`.

NOTE

This feature is **disabled** by default if using a `tsconfig.json` configuration file, but may be set to enabled as outlined further below.

Currently auto-detection works for dependencies downloaded from npm (by reading the `package.json` file), Bower (by reading the `bower.json` file), and for loose files in your project that match a list of roughly the top 400 most popular JavaScript libraries. For example, if you have `jquery-1.10.min.js` in your project, the file `jquery.d.ts` will be fetched and loaded in order to provide a better editing experience. This `.d.ts` file will have no impact on your project.

If you do not wish to use auto-acquisition, disable it by adding a configuration file as outlined below. You can still place definition files for use directly within your project manually.

See also

- [Using IntelliSense](#)
- [JavaScript support \(Visual Studio for Mac\)](#)

Visual C++ IntelliSense features

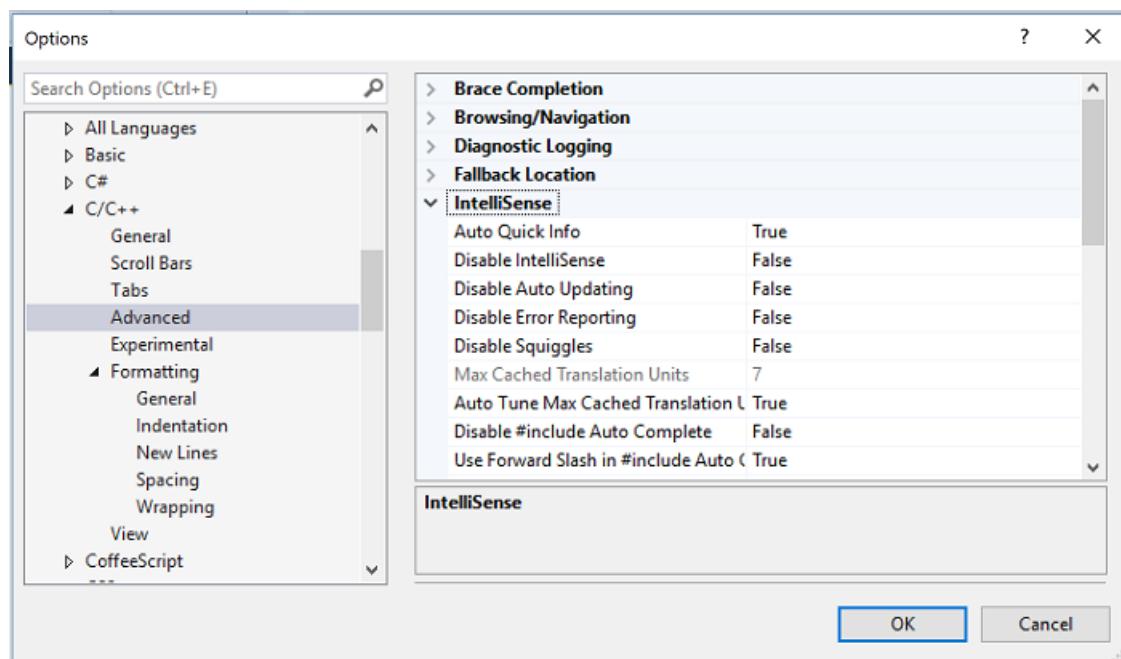
7/24/2019 • 5 minutes to read • [Edit Online](#)

IntelliSense is a name given to a set of features that make coding more convenient. IntelliSense for C++ is available for stand-alone files as well as for files that are part of a C++ project. In cross-platform projects, some IntelliSense features are available in .cpp and .c files in the shared code project, even when you are in an Android or iOS context.

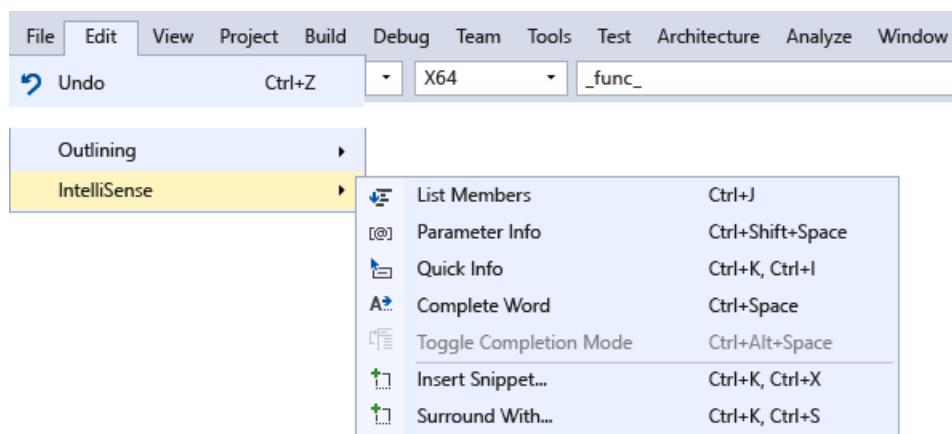
This article provides an overview of C++ IntelliSense features. For information on how to configure your project for IntelliSense and how to troubleshoot problems, see [Configure a C++ project for IntelliSense](#).

IntelliSense features in C++

IntelliSense is a name given to a set of features that make coding more convenient. Since different people have different ideas about what is convenient, virtually all of the IntelliSense features can be enabled or disabled in the **Options** dialog box, under **Text Editor > C/C++ > Advanced**. The **Options** dialog box is available from the **Tools** menu on the menu bar.



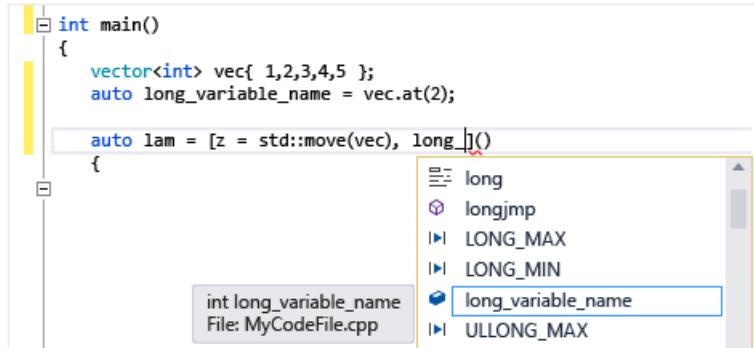
You can use the menu items and keyboard shortcuts shown in the following image to access IntelliSense.



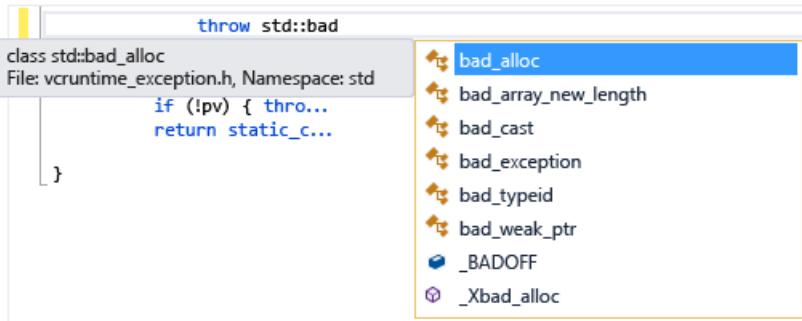
Statement completion and member list

When you start typing a keyword, type, function, variable name, or other program element that the compiler recognizes, the editor offers to complete the word for you.

For a list of the icons and their meanings, see [Class View and Object Browser icons](#).

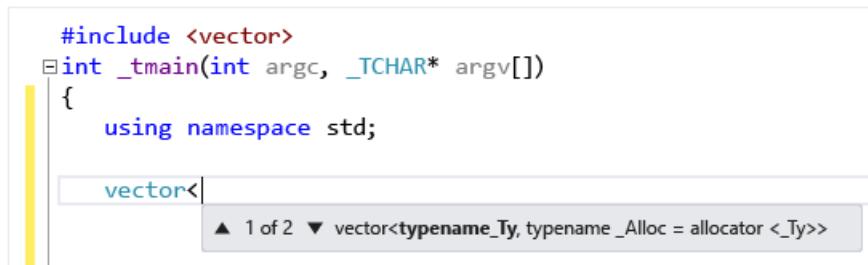


The first time that you invoke member list, it only shows members that are accessible for the current context. If you press **Ctrl+J** after that, it shows all members regardless of accessibility. If you invoke it a third time, an even wider list of program elements is shown. You can turn off member list in the **Options** dialog box, under **Text Editor > C/C++ > General > Auto list members**.



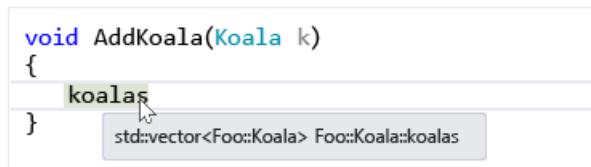
Parameter help

When you type an opening brace of a function call, or angle bracket on a class template variable declaration, the editor shows a small window with the parameter types for each overload of the function or constructor. The "current" parameter—based on the cursor location—is in bold. You can turn off parameter information in the **Options** dialog box, under **Text Editor > C/C++ > General > Parameter information**.



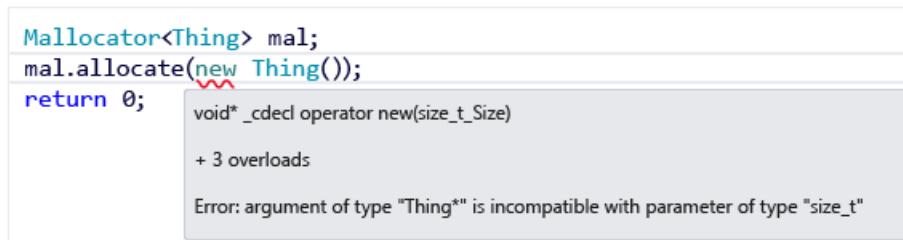
Quick Info

When you hover the mouse cursor over a variable, a small window appears inline that shows the type information and the header in which the type is defined. Hover over a function call to see the function's signature. You can turn off Quick Info in the **Options** dialog box, under **Text Editor > C/C++ > Advanced > Auto Quick Info**.



Error squiggles

Squiggles under a program element (variable, keyword, brace, type name, and so on) call your attention to an error or potential error in the code. A green squiggle appears when you write a forward declaration, to remind you that you still need to write the implementation. A purple squiggle appears in a shared project when there is an error in code that is not currently active, for example when you are working in the Windows context but enter something that would be an error in an Android context. A red squiggle indicates a compiler error or warning in active code that you need to deal with.



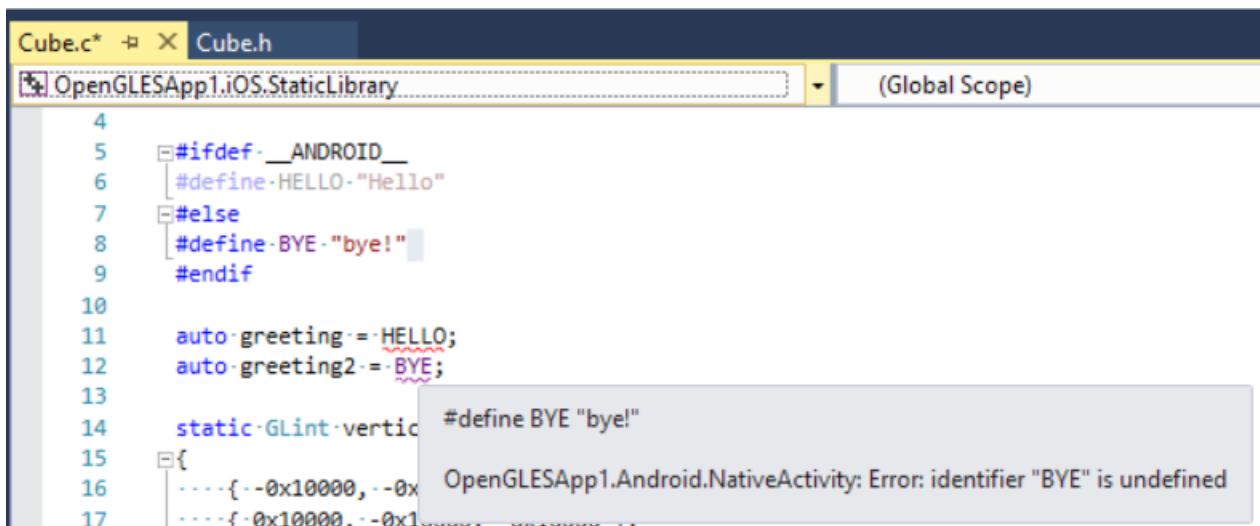
Code colorization and fonts

The default colors and fonts can be changed in the **Options** dialog box, under **Environment > Fonts and Colors**. You can change the fonts for many UI windows here, not just the editor. The settings that are specific to C++ begin with "C++"; the other settings are for all languages.

Cross-platform IntelliSense

In a shared code project, some IntelliSense features such as squiggles are available even when you are working in an Android context. If you write some code that would result in an error in an inactive project, IntelliSense still shows squiggles, but they are in a different color than squiggles for errors in the current context.

Consider an OpenGLES Application that's configured to build for Android and iOS. The illustration shows shared code being edited. In this image, the active project is **iOS.StaticLibrary**:



Notice the following:

- The `#ifdef` branch on line 6 is grayed out to indicate an inactive region, because `__ANDROID__` is not defined for the iOS project.

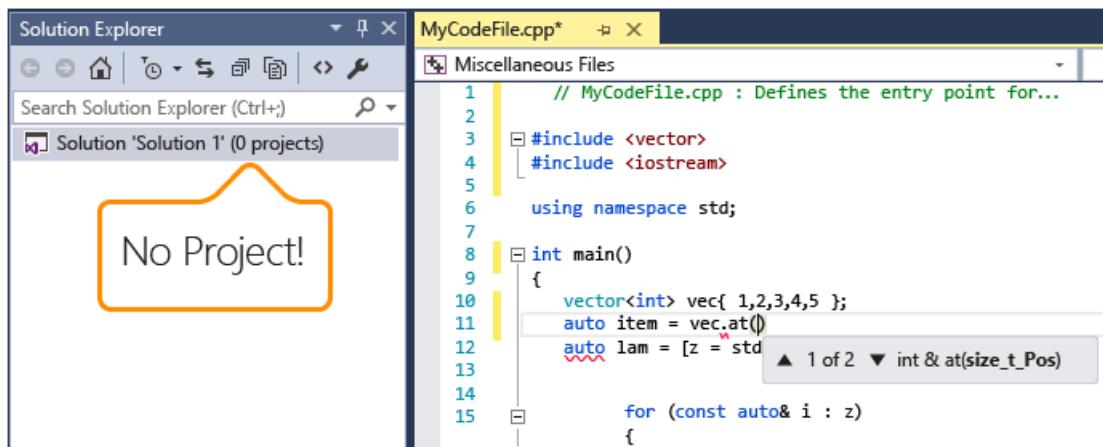
- The greeting variable at line 11 is initialized with the identifier `HELLO`, which now has a red squiggle. This is because no identifier `HELLO` is defined in the currently active iOS project.
- Line 12 has a purple squiggle on the identifier `BYE` because this identifier isn't defined in the (currently) inactive **Android.NativeActivity** project. Even though this line compiles when iOS is the active project, it won't compile when Android is the active project. Since this is shared code, you should correct the code even though it compiles in the currently active configuration.

If you change the active project to Android, the squiggles change:

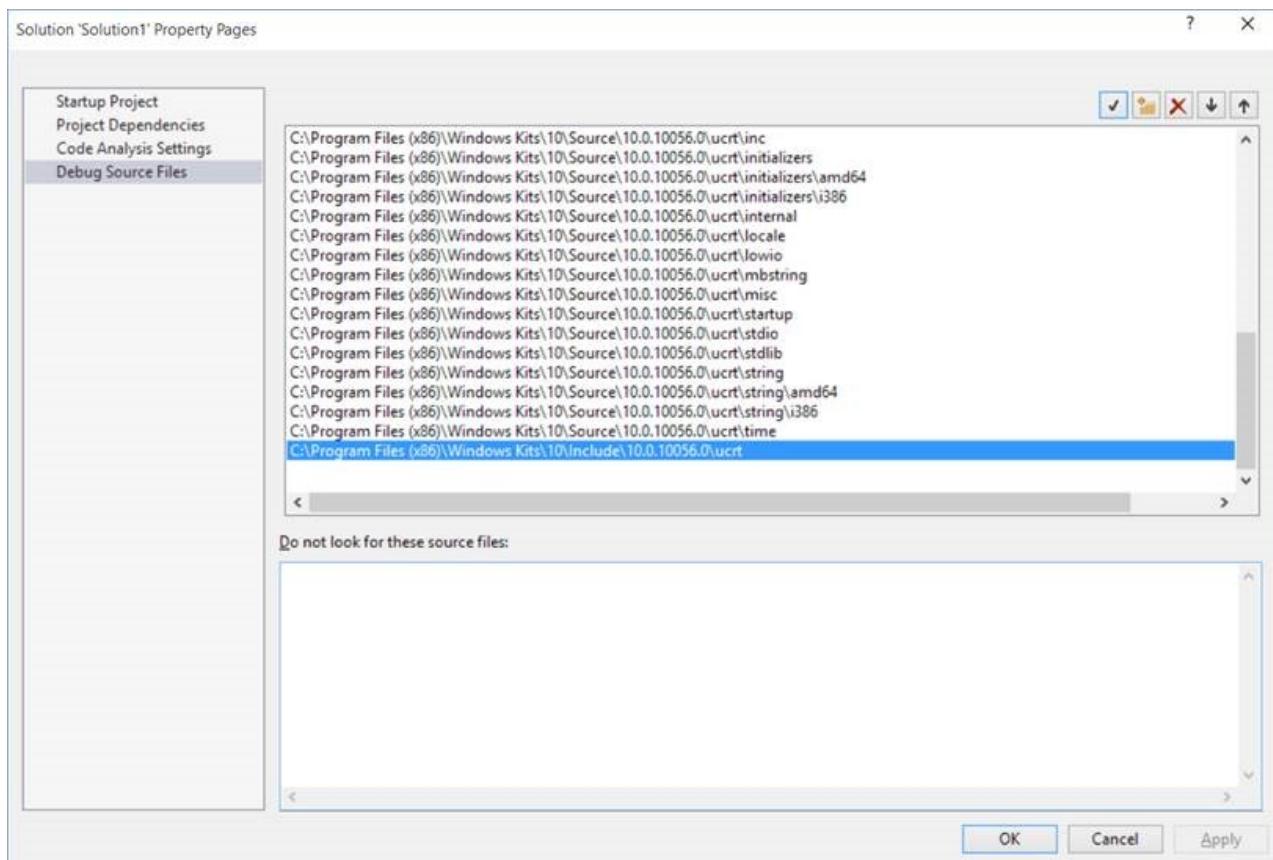
- The `#else` branch on line 8 is grayed out to indicate an inactive region, because `__ANDROID__` is defined for Android project.
- The greeting variable at line 11 is initialized with identifier `HELLO`, which has a purple squiggle. This is because no identifier `HELLO` is defined in the currently inactive iOS project.
- Line 12 has a red squiggle on the identifier `BYE` because this identifier is not defined in the active project.

IntelliSense for stand-alone files

When you open a single file outside of any project, you still get IntelliSense. You can enable or disable particular IntelliSense features in the **Options** dialog box, under **Text Editor > C/C++ > Advanced**. To configure IntelliSense for single files that aren't part of a project, look for the **IntelliSense and browsing for non-project files** section.

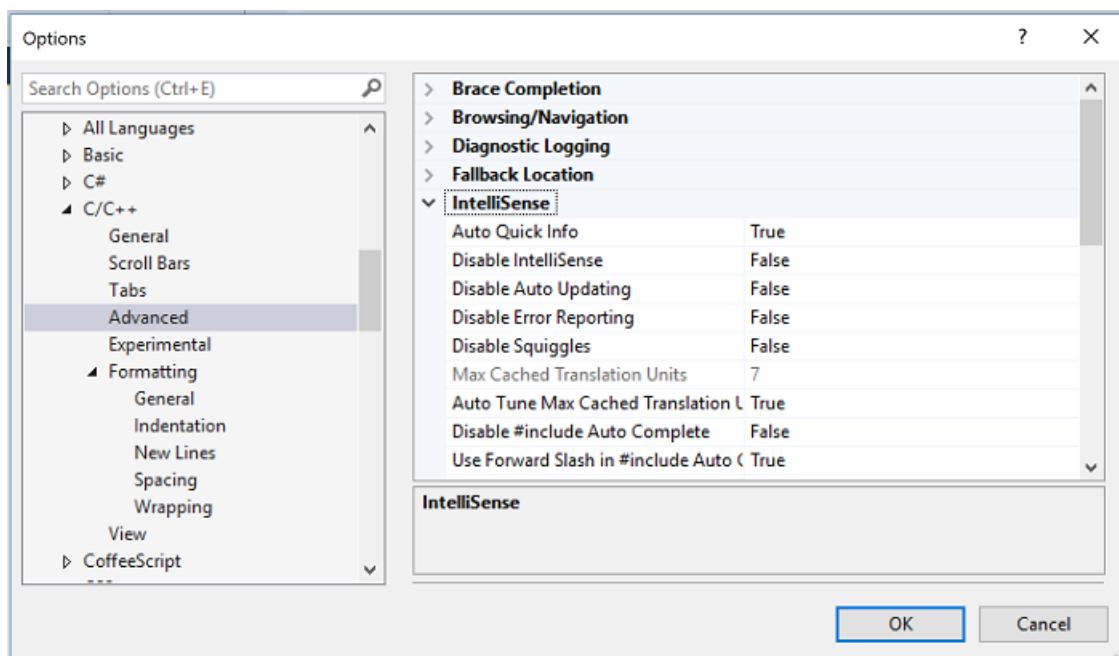


By default, single file IntelliSense only uses standard include directories to find header files. To add additional directories, open the shortcut menu on the **Solution** node, and add your directory to **Debug Source Code** list, as the following illustration shows:



Enable or disable features

Since different people have different ideas about what is convenient, virtually all of the IntelliSense features can be enabled or disabled in the **Options** dialog box, under **Text Editor > C/C++ > Advanced**. The **Options** dialog box is available from the **Tools** menu on the menu bar.



See also

- [Using IntelliSense](#)
- [Configure a C++ project for IntelliSense](#)

Configure a C++ project for IntelliSense

10/21/2019 • 5 minutes to read • [Edit Online](#)

In some cases, you might need to manually configure your C++ project to get IntelliSense working properly. For MSBuild projects (based on .vcxproj files), you can adjust settings in project properties. For non-MSBuild projects, you adjust settings in the CppProperties.json file in the root directory of the project. In some cases, you may need to create a hint file to help IntelliSense understand macro definitions. The Visual Studio IDE helps you identify and fix IntelliSense problems.

Single-file IntelliSense

When you open a file that is not included in a project, Visual Studio provides some IntelliSense support but by default no error squiggles are shown. If the **Navigation Bar** says *Miscellaneous Files*, then that probably explains why you are not seeing error squiggles under incorrect code, or why a preprocessor macro is not defined.

Check the Error List

If a file is not open in single-file mode, and IntelliSense is not working correctly, the first place to check is the Error List window. To see all the IntelliSense errors for the current source file together with all included header files, choose **Build + IntelliSense** in the dropdown:

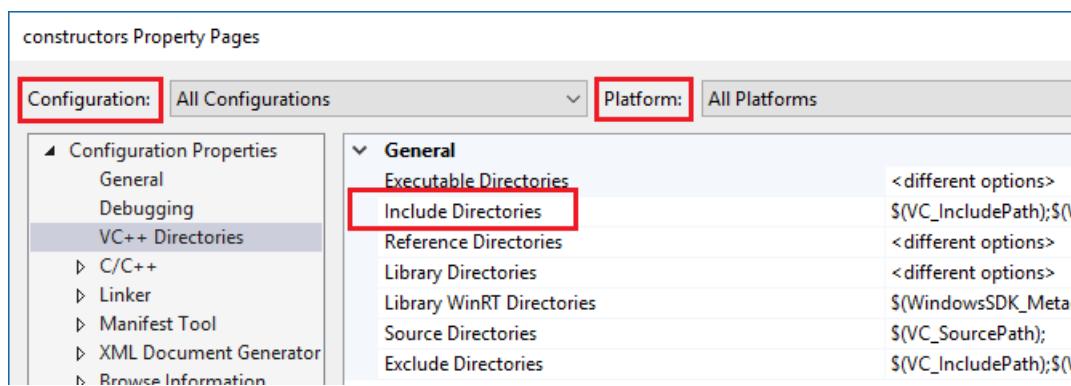


IntelliSense produces a maximum of 1000 errors. If there are over 1000 errors in the header files included by a source file, then the source file shows only a single error squiggle at the very start of the source file.

Ensure #include paths are correct

MSBuild projects

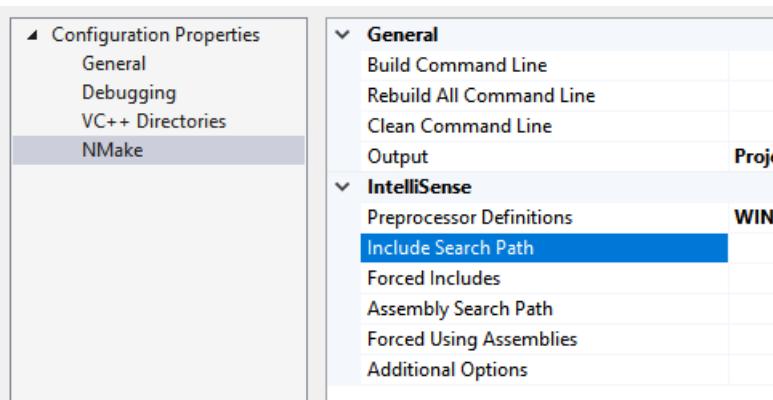
If you run your builds outside of the Visual Studio IDE, and your builds are succeeding but IntelliSense is incorrect, it is possible that your command line is out of sync with the project settings for one or more configurations. Right-click on the project node in **Solution Explorer** and make sure that all **#include** paths are correct for the current configuration and platform. If the paths are identical in all configurations and platforms, you can select **All configurations** and **All platforms** and then verify that the paths are correct.



To see the current values for build macros such as **VC_IncludePath**, select the Include Directories line and click the dropdown on the right. Then choose **<Edit>** and click on the **Macros** button.

Makefile projects

For Makefile projects that are based on the NMake project template, choose **NMake** in the left pane and then choose **Include search path** under the **IntelliSense** category:



Open Folder projects

For CMake projects, make sure that #include paths are specified correctly for all configurations in CMakeLists.txt. Other project types might require a CppProperties.json file. For more information, see [Configure IntelliSense with CppProperties.json](#). Make sure that the paths are correct for each configuration that is defined in the file.

If there is a syntax error in the CppProperties.json file, IntelliSense in the affected files will be incorrect. Visual Studio will display the error in the Output Window.

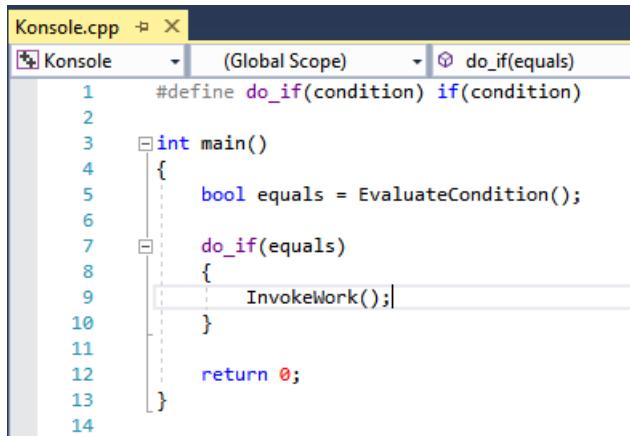
Tag parser issues

The tag parser is a "fuzzy" C++ parser that is used for browsing and navigation. It is very fast but does not attempt to completely comprehend every code construct.

For example, it doesn't evaluate preprocessor macros, and therefore it may incorrectly parse code that makes heavy use of them. When the Tag Parser encounters an unfamiliar code construct, it may skip that entire region of code.

There are two common ways in which this problem manifests in Visual Studio:

1. If the Navigation Bar shows an innermost macro, then the current function definition was skipped:



2. The IDE offers to create a function definition for a function that is already defined:

```
class Class
{
    int Function();
};

int Class::Function()
{
    bool equals = EvaluateCondition(); do_if>equals)
    {
        InvokeWork();
    }

    return 0;
}
```

To fix these kinds of problems, add a file named **cpp.hint** to the root of your solution directory. For more information, see [Hint Files](#).

Tag parser errors appear in the **Error List** window.

Validate project settings with diagnostic logging

To check whether IntelliSense compiler is using correct compiler options, including Include Paths and Preprocessor macros, turn on Diagnostic Logging of IntelliSense command lines in **Tools > Options > Text Editor > C/C++ > Advanced > Diagnostic Logging**. Set **Enable Logging** to True, **Logging Level** to 5 (most verbose), and **Logging Filter** to 8 (IntelliSense logging).

The Output Window will now show the command lines that are passed to the IntelliSense compiler. Here is a sample output:

```
[IntelliSense] Configuration Name: Debug|Win32
[IntelliSense] Toolset IntelliSense Identifier:
[IntelliSense] command line options:
/c
/I.
/IC:\Repo\Includes
/DWIN32
/DDEBUG
/D_DEBUG
/Zc:wchar_t-
/Zc:forScope
/Yustdafx.h
```

This information may help you understand why IntelliSense is providing inaccurate information. For example, if your project's Include directory contains **\$(MyVariable)\Include**, and the diagnostic log shows **/I\Include** as an Include path, it means that **\$(MyVariable)** wasn't evaluated, and was removed from the final include path.

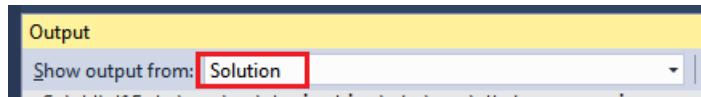
About the IntelliSense build

Visual Studio uses a dedicated C++ compiler to create and maintain the database that powers all the IntelliSense features. To keep the IntelliSense database in sync with the code, Visual Studio automatically launches IntelliSense-only builds as background tasks in response to certain changes made in the project settings or source files.

However, in some cases Visual Studio might not update the IntelliSense database in a timely manner. For example, when you run a **git pull** or **git checkout** command, Visual Studio might take up to an hour to detect changes in the files. You can force a rescan of all files in a solution by right-clicking on the project node in **Solution Explorer** and choosing **Rescan Solution**.

Troubleshooting IntelliSense build failures

An IntelliSense build does not produce binaries, but it can still fail. One possible cause for failure is custom .props or .targets files. In Visual Studio 2017 version 15.6 and later, IntelliSense-only build errors are logged to the Output window. To see them, set **Show output from** to **Solution**:



The error message might instruct you to enable design-time tracing:

```
error: Designtime build failed for project 'E:\src\MyProject\MyProject.vcxproj',
configuration 'Debug|x64'. IntelliSense might be unavailable.
Set environment variable TRACEDESIGNTIME=true and restart
Visual Studio to investigate.
```

If you set the environment variable TRACEDESIGNTIME to true and restart Visual Studio, you will see a log file in the %TEMP% directory, which might help diagnose the build failure.

To learn more about TRACEDESIGNTIME environment variable, see [Roslyn](#) and [Common Project System](#). The information in these articles is relevant for C++ projects.

See also

- [Visual C++ IntelliSense](#)

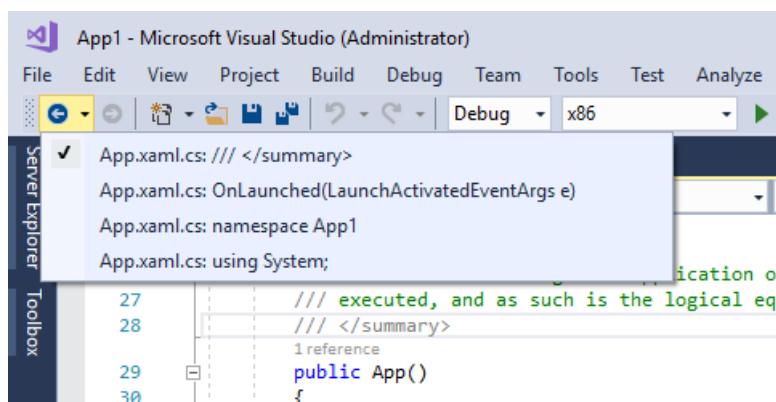
Navigate code

11/22/2019 • 6 minutes to read • [Edit Online](#)

Visual Studio provides numerous ways to navigate code in the editor. This topic summarizes the different ways you can navigate your code, and provides links to topics that go into more detail.

Navigate Backward and Navigate Forward commands

You can use the **Navigate Backward** (**Ctrl+--**) and **Navigate Forward** (**Ctrl+Shift+--**) buttons on the toolbar to move the insertion point to previous locations, or to return to a more recent location from a previous location. These buttons retain the last 20 locations of the insertion point. These commands are also available on the **View** menu, under **Navigate Backward** and **Navigate Forward**.



Navigation bar

You can use the **navigation bar** (the drop-down boxes at the top of the code window) to navigate to code in a codebase. You can choose a type or member to go directly to it. The navigation bar appears when you edit code in a Visual Basic, C#, or C++ code base. In a partial class, members defined outside the current code file may be disabled (they appear in gray).

You can navigate around the drop-down boxes as follows:

- To navigate to another project that the current file belongs to, choose it in the left drop-down.
- To navigate to a class or type, choose it in the middle drop-down.
- To navigate directly to a procedure or other member of a class, choose it in the right drop-down.
- To shift focus from the code window to the navigation bar, press the shortcut key combination **Ctrl+F2**.
- To shift focus from box to box on the navigation bar, press the **Tab** key.
- To select the navigation bar item that has focus and return to the code window, press the **Enter** key.
- To return focus from the navigation bar to the code without selecting anything, press the **Esc** key.

To hide the navigation bar, change the **Navigation bar** option in the **Text Editor All Languages** settings (**Tools > Options > Text Editor > All Languages**), or you can change the settings for individual languages.

Find all references

Finds all the references to the selected element in the solution. You can use this to check possible side-effects of a large refactoring, or to verify "dead" code. Press **F8** to jump between results. For more information, see [Find references in your code](#).

INPUT	FUNCTION
Keyboard	Place your text cursor somewhere inside the type name, and press Shift+F12
Mouse	Select Find All References from the right-click menu

Reference highlighting

When you click a symbol in the source code, all instances of that symbol are highlighted in the document. The highlighted symbols may include declarations and references, and many other symbols that [Find All References](#)

would return. These include the names of classes, objects, variables, methods, and properties. In Visual Basic code, keywords for many control structures are also highlighted. To move to the next or the previous highlighted symbol, press **Ctrl+Shift+Down Arrow** or **Ctrl+Shift+Up Arrow**. You can change the highlighting color in **Tools > Options > Environment > Fonts and Colors > Highlighted Reference**.

Go To commands

Go To has the following commands, which are available in the **Edit** menu under **Go To**:

- **Go To Line (Ctrl+G)**: Move to the specified line number in the active document.
- **Go To All (Ctrl+T or Ctrl+,)**: Move to the specified line, type, file, member, or symbol.
- **Go To File (Ctrl+1, Ctrl+F)**: Move to the specified file in the solution.
- **Go To Recent File (Ctrl+1, Ctrl+R)**: Move to the specified, recently visited file in the solution.
- **Go To Type (Ctrl+1, Ctrl+T)**: Move to the specified type in the solution.
- **Go To Member (Ctrl+1, Ctrl+M)**: Move to the specified member in the solution.
- **Go To Symbol (Ctrl+1, Ctrl+S)**: Move to the specified symbol in the solution.

In Visual Studio 2017 version 15.8 and later, the following **Go To** navigation commands are also available:

- **Go To Next Issue in File (Alt+PgDn)** and **Go To Previous Issue in File (Alt+PgUp)**
- **Go To Last Edit Location (Ctrl+Shift+Backspace)**

See more about these commands in the [Find code using Go To commands](#) topic.

Go To Definition

Go To Definition takes you to the definition of the selected element. For more information, see [Go To Definition and Peek Definition](#).

INPUT	FUNCTION
Keyboard	Place your text cursor somewhere inside the type name, and press F12
Mouse	Right-click on the type name and select Go To Definition OR press Ctrl and click on the type name

Peek Definition

Peek Definition displays the definition of the selected element in a window without navigating away from your current location in the code editor. For more information, see [How to: View and edit code by using Peek Definition](#) and [Go To Definition and Peek Definition](#).

INPUT	FUNCTION
Keyboard	Place your text cursor somewhere inside the type name, and press Alt+F12
Mouse	Right-click on the type name and select Peek Definition OR press Ctrl and click on the type name (if you have the Open definition in peek view option checked)

Go To Implementation

Using Go To Implementation, you can navigate from a base class or type to its implementations. If there are multiple implementations, you will see them listed in the **Find Symbol Results** window:

INPUT	FUNCTION
Keyboard	Place your text cursor somewhere inside the type name, and press Ctrl+F12
Mouse	Right-click on the type name and select Go To Implementation

Go To Base

Using Go To Base, you can navigate up the inheritance chain of the selected element. If there are multiple results, you will see them listed in the **Go To Base** window:

INPUT	FUNCTION
Keyboard	Place your text cursor somewhere inside the type name, and press Alt+Home
Mouse	Right-click on the type name and select Go To Base

Call Hierarchy

You can view calls to and from a method in the [Call Hierarchy window](#):

INPUT	FUNCTION
Keyboard	Place your text cursor somewhere inside the type name, and press Ctrl+K, Ctrl+T
Mouse	Right-click on the member name and select View Call Hierarchy

Next Method and Previous Method commands (Visual Basic)

In Visual Basic code files, use these commands to move the insertion point to different methods. Choose **Edit > Next Method** or **Edit > Previous Method**.

Structure Visualizer

The Structure Visualizer feature in the code editor shows *structure guide lines* - vertical dashed lines that indicate matching curly braces in your codebase. This makes it easier to see where logical blocks begin and end.

The screenshot shows a code editor window for a file named 'NavigationService.cs'. The code is written in C# and defines a method 'InternalNavigateToAsync' that takes a type and an object as parameters. The code uses several if statements to handle different types of pages (MainPage, LoginPage, CustomNavigationPage). Red boxes highlight two specific sections of the code: one around the first if statement where 'CurrentApplication.MainPage' is assigned, and another around the else block where a new 'CustomNavigationPage' is created and its 'Detail' property is set to the current page. The code editor interface includes a top bar with tabs for 'Miscellaneous Files', 'BikeSharing.Clients.Core.Services.Nav', and '_authenticationService'. On the left, there's a vertical structure guide line for the code blocks. The right side features a vertical scroll bar.

```
95     protected virtual async Task InternalNavigateToAsync(Type viewModelType, obj
96     {
97         Page page = CreateAndBindPage(viewModelType, parameter);
98
99         if (page is MainPage)
100        {
101            CurrentApplication.MainPage = page;
102        }
103        else if (page is LoginPage)
104        {
105            CurrentApplication.MainPage = new CustomNavigationPage(page);
106        }
107        else if (CurrentApplication.MainPage is MainPage)
108        {
109            var mainPage = CurrentApplication.MainPage as MainPage;
110            var navigationPage = mainPage.Detail as CustomNavigationPage;
111
112            if (navigationPage != null)
113            {
114                await navigationPage.PushAsync(page);
115            }
116            else
117            {
118                navigationPage = new CustomNavigationPage(page);
119                mainPage.Detail = navigationPage;
120
121                mainPage.IsPresented = false;
122            }
123        }
124    }
```

To disable structure guide lines, go to **Tools > Options > Text Editor > General** and clear the **Show structure guide lines** box.

Enhanced scroll bar

You can use the enhanced scroll bar in a code window to get a bird's-eye view of your code. In map mode, you can see previews of the code when you move the cursor up and down the scroll bar. For more information, see [How to: Track your code by customizing the scroll bar](#).

CodeLens information

You can find info about specific code, like changes and who made those changes, references, bugs, work items, code reviews, and unit test status when you use CodeLens in the code editor. CodeLens works like a heads-up display when you use Visual Studio Enterprise with Team Foundation Server. See [Find code changes and other history](#).

See also

- [Features of the code editor](#)
- [View call hierarchy](#)

Find references in your code

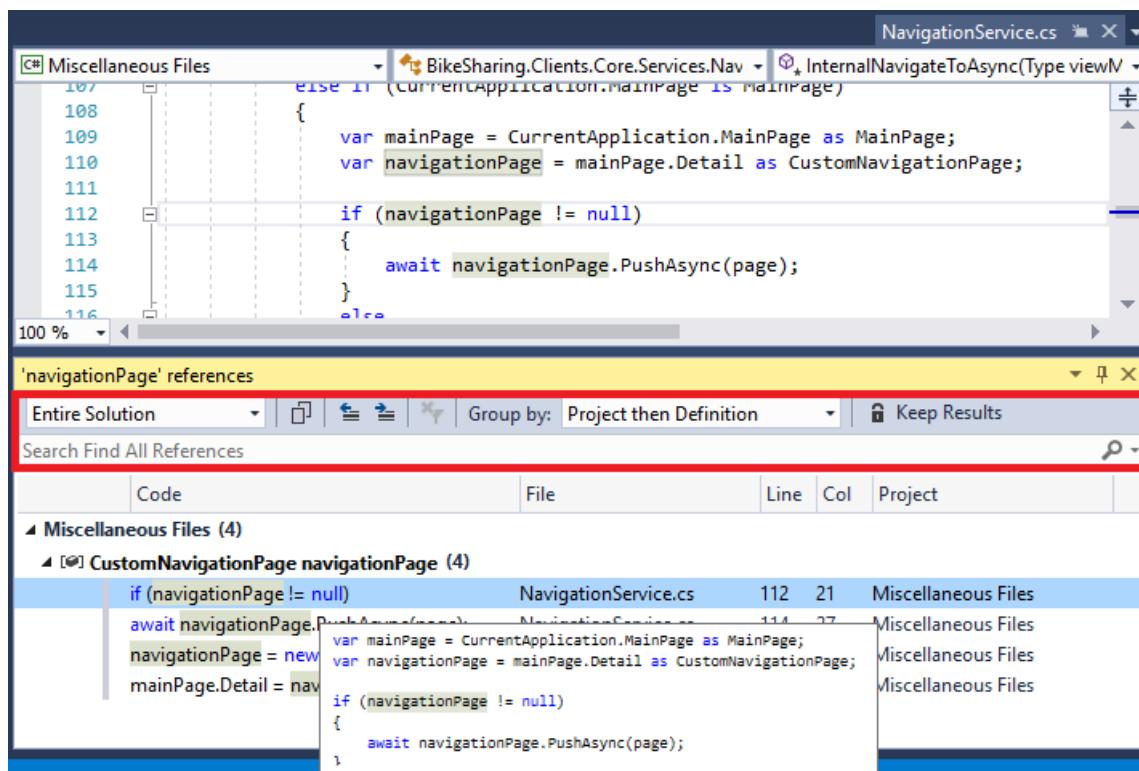
10/18/2019 • 2 minutes to read • [Edit Online](#)

You can use the **Find All References** command to find where particular code elements are referenced throughout your codebase. The **Find All References** command is available on the context (right-click) menu of the element you want to find references to. Or, if you are a keyboard user, press **Shift + F12**.

The results appear in a tool window named **<element> references**, where *element* is the name of the item you are searching for. A toolbar in the **references** window enables you to:

- Change the scope of the search in a drop-down list box. You can choose to look only in changed documents all the way up to the entire solution.
- Copy the selected referenced item by choosing the **Copy** button.
- Choose buttons to go to the next or previous location in the list, or press the **F8** and **Shift + F8** keys to do so.
- Remove any filters on the returned results by choosing the **Clear All Filters** button.
- Change how returned items are grouped by choosing a setting in the **Group by:** drop-down list box.
- Keep the current search results window by choosing the **Keep Results** button. When you choose this button, the current search results stay in this window, and new search results appear in a new tool window.
- Search for strings within the search results by entering text in the **Search Find All References** text box.

You can also hover the mouse over any search result to see a preview of the reference.



Navigate to references

You can use the following methods to navigate to references in the **references** window:

- Press **F8** to go to the next reference, or **Shift + F8** to go to the previous reference.
- Press the **Enter** key on a reference, or double-click it, to go to it in code.
- On the right-click menu (context menu) of a reference, choose the **Go To Previous Location** or **Go To Next**

Location commands.

- Choose the **Up Arrow** and **Down Arrow** keys (if they are enabled in the **Options** dialog box). To enable this functionality, on the menu bar, choose **Tools > Options > Environment > Tabs and Windows > Preview Tab**, and then select the **Allow new files to be opened in the preview tab** and **Preview selected files in Find Results** boxes.

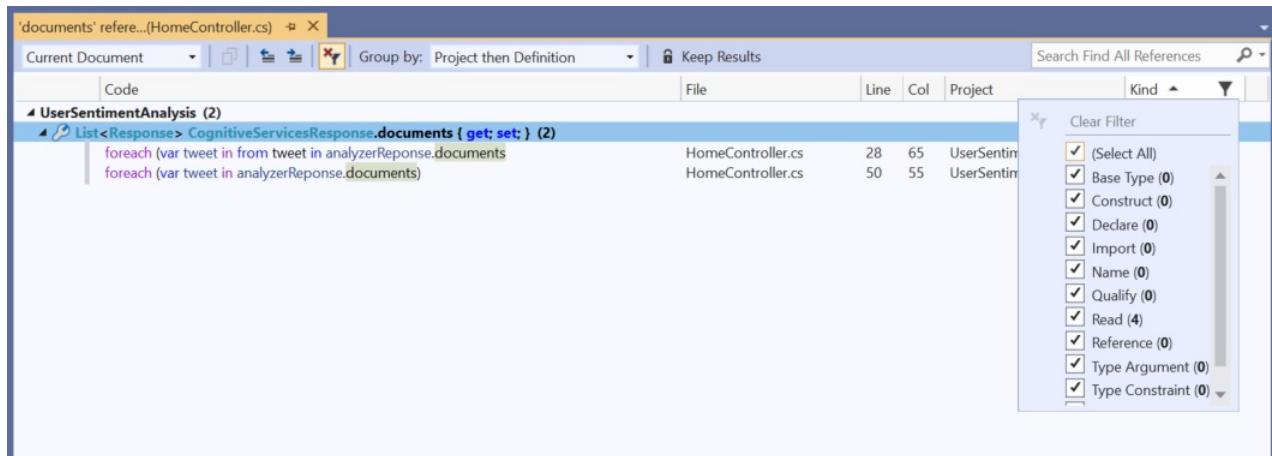
Change reference groupings

By default, references are grouped by project, then by definition. However, you can change this grouping order by changing the setting in the **Group by:** drop-down list box on the toolbar. For example, you can change it from the default setting of **Project then definition** to **Definition then project**, as well to other settings.

Definition and **Project** are the two default groupings used, but you can add others by choosing the **Grouping** command on the selected item's right-click or context menu. Adding more groupings can be helpful if your solution has a lot of files and paths.

Filter by reference type in .NET

In C# or Visual Basic, the Find References window has a Kind column where it lists what type of reference it found. This column can be used to filter by reference type by clicking on the filter icon that appears when hovering over the column header. References can be filtered by Read, Write, Reference, Name, Namespace, and Type.



See also

- [Navigating code](#)

View type and member definitions

10/18/2019 • 3 minutes to read • [Edit Online](#)

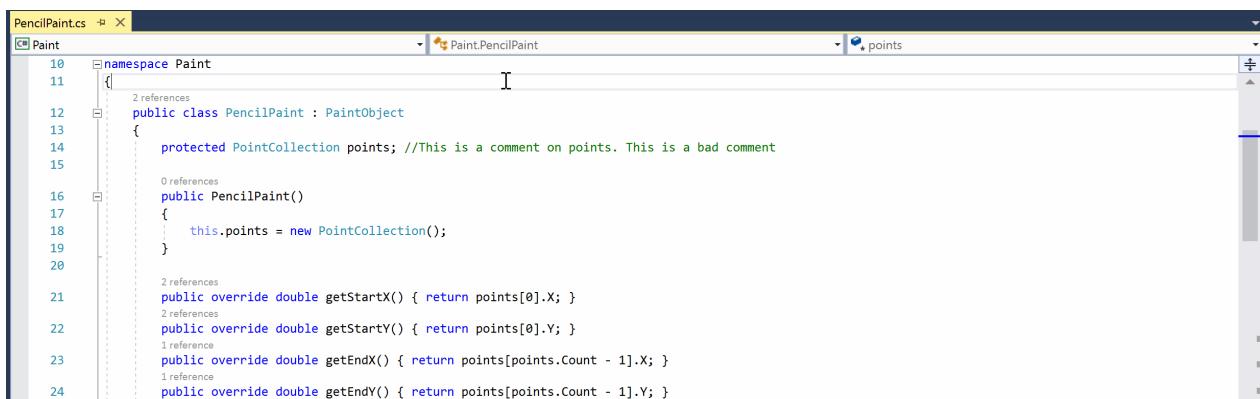
Developers often need to view the source code definitions for types or class members they use in their code. In Visual Studio, the **Go To Definition** and **Peek Definition** features enable you to easily view the definition of a type or member. If the source code is not available, metadata is displayed instead.

Go To Definition

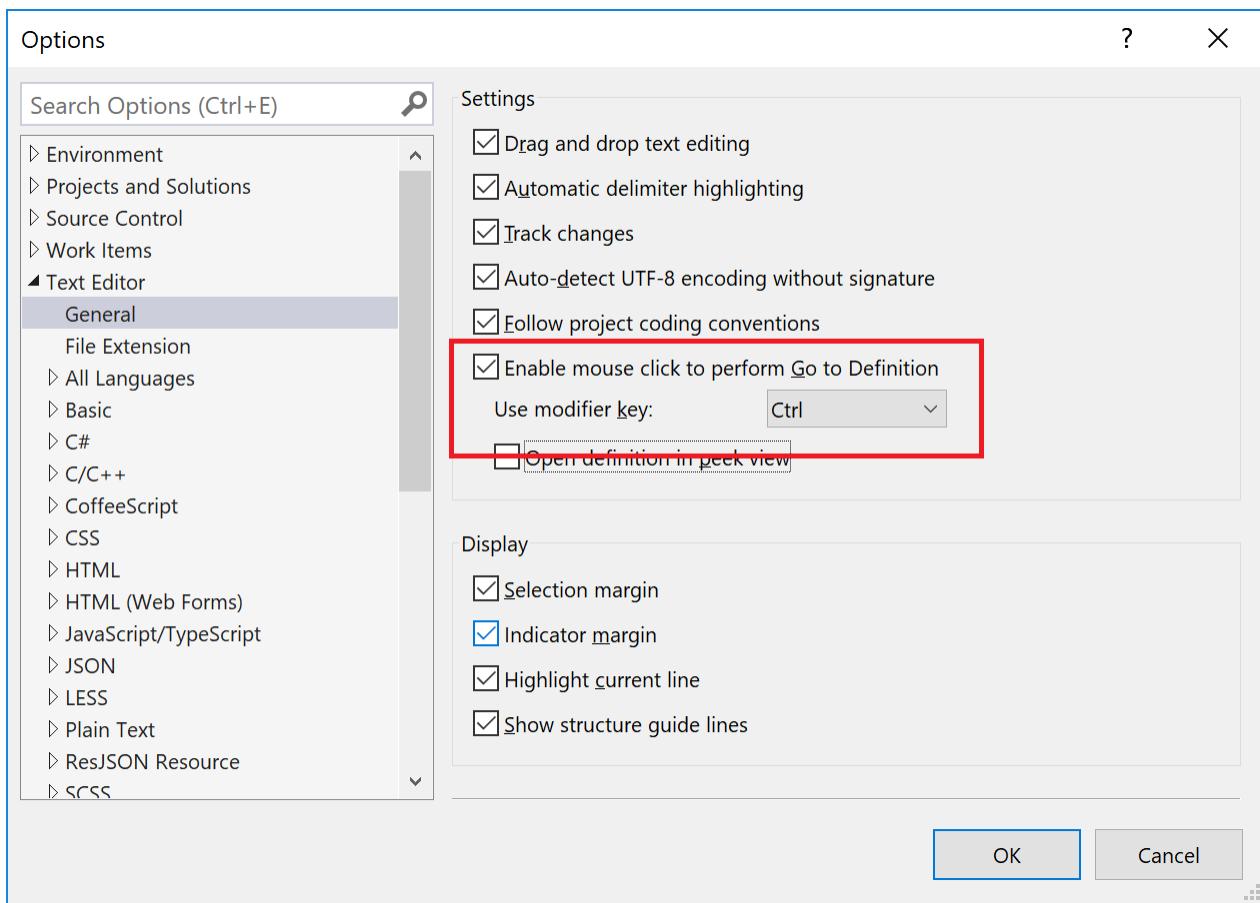
The **Go To Definition** feature navigates to the source of a type or member, and opens the result in a new tab. If you are a keyboard user, place your text cursor somewhere inside the symbol name and press **F12**. If you are a mouse user, either select **Go To Definition** from the right-click menu or use the **Ctrl-click** functionality described in the following section.

Ctrl-click Go To Definition

Ctrl+click is a shortcut for mouse users to quickly access **Go To Definition**. Symbols become clickable when you press **Ctrl** and hover over the type or member. To quickly navigate to the definition of a symbol, press the **Ctrl** key and then click on it. It's that easy!



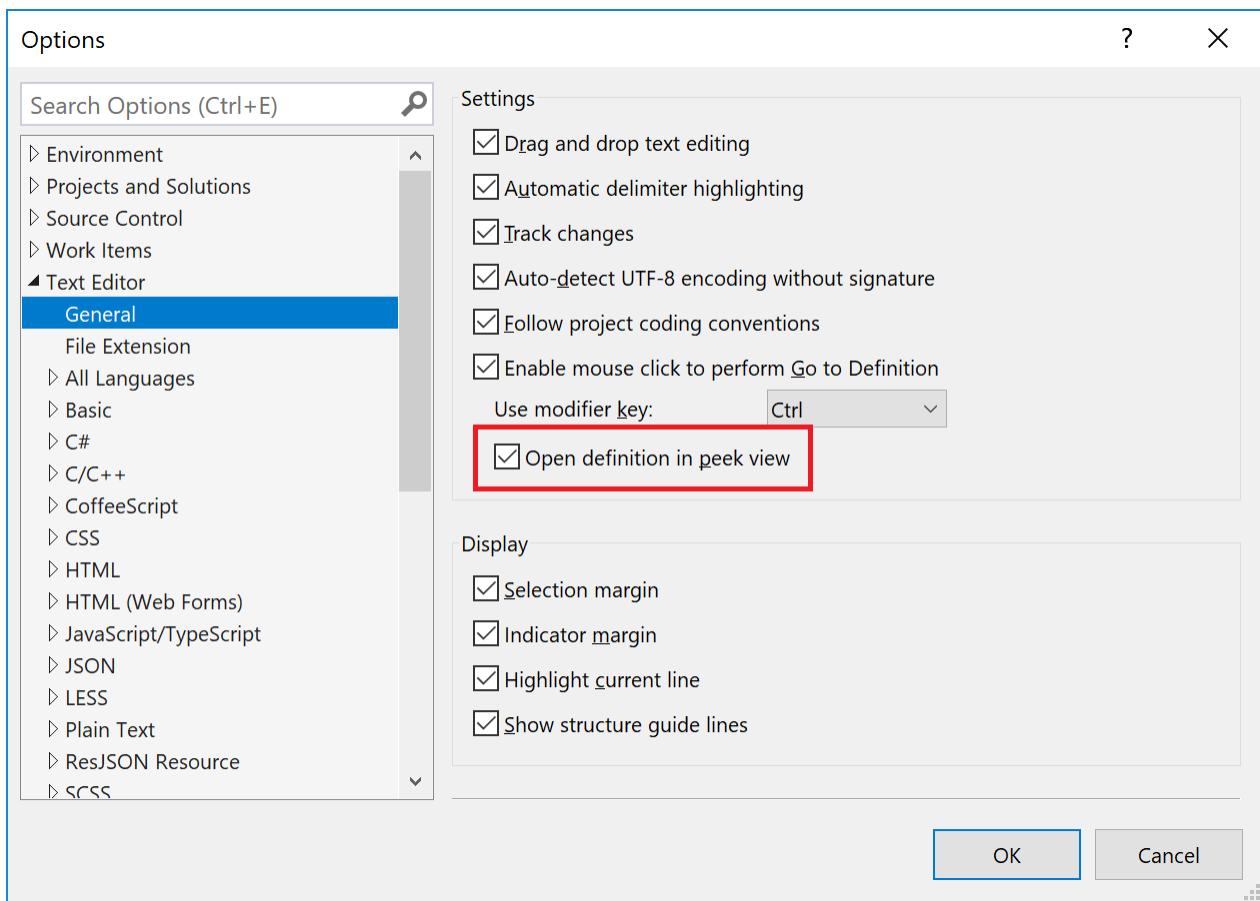
You can change the modifier key for mouse-click **Go To Definition** by going to **Tools > Options > Text Editor > General**, and selecting either **Alt** or **Ctrl+Alt** from the **Use modifier key** drop-down. You can also disable mouse-click **Go To Definition** by unchecking the **Enable mouse click to perform Go To Definition** checkbox.



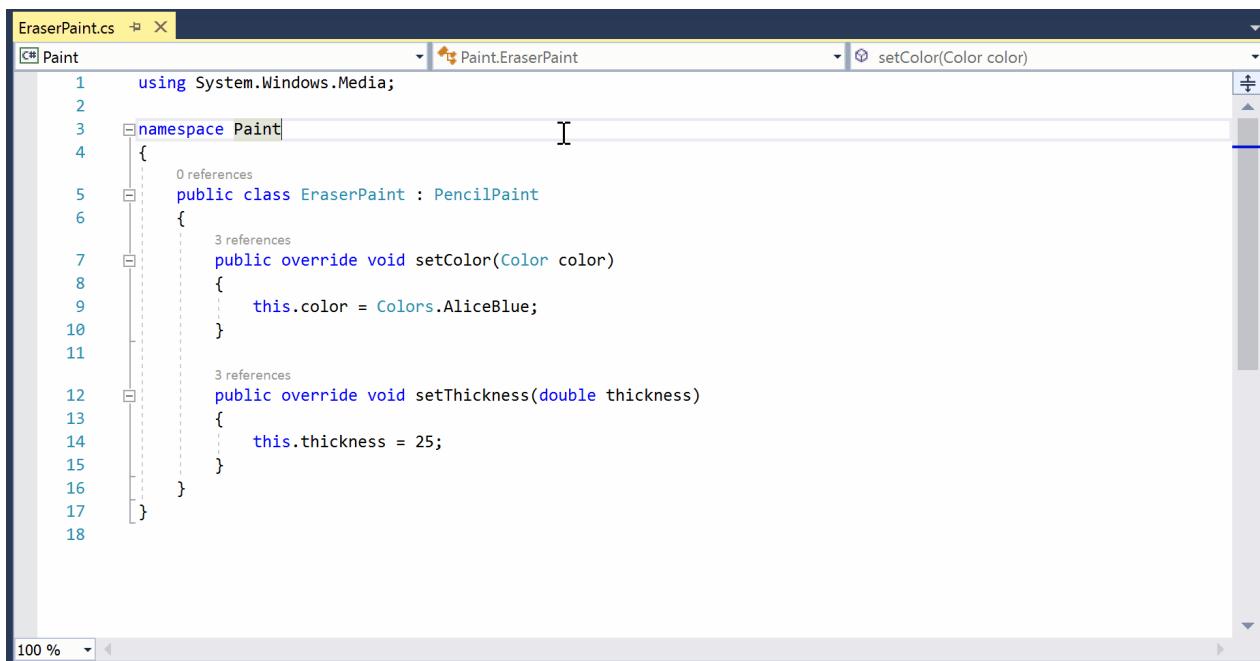
Peek Definition

The **Peek Definition** feature lets you preview the definition of a type without leaving your current location in the editor. If you are a keyboard user, place your text cursor somewhere inside the type or member name and press **Alt + F12**. If you are a mouse user, you can select **Peek Definition** from the right-click menu.

To enable **Ctrl+click** functionality, go to **Tools > Options > Text Editor > General**. Select the option **Open definition in peek view** and click **OK** to close the **Options** dialog box.



Then, press **Ctrl** (or whichever modifier key is selected in **Options**), and click on the type or member.



If you peek another definition from the popup window, you start a breadcrumb path that you can navigate using the circles and arrows that appear above the popup.

For more information, see [How to: View and edit code by using Peek Definition \(Alt+F12\)](#).

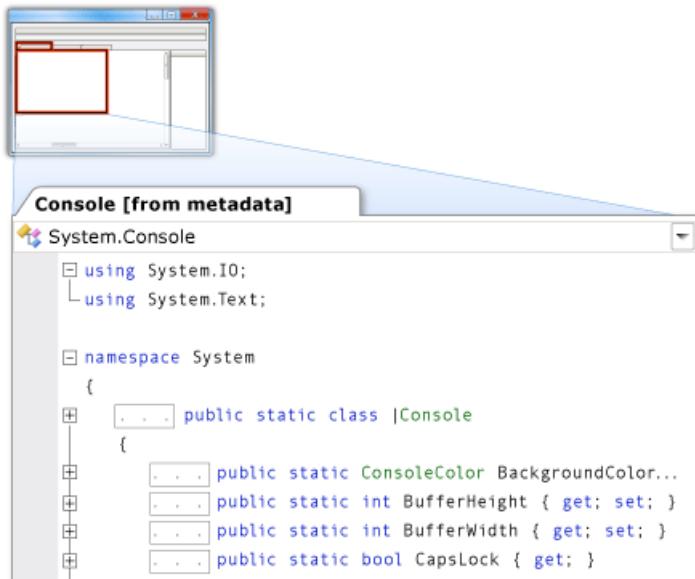
View metadata as source code (C#)

When you view the definition of C# types or members whose source code is not available, their metadata is displayed instead. You can see the declarations of the types and members, but not their implementations.

When you run the **Go To Definition** or **Peek Definition** command for an item whose source code is

unavailable, a tabbed document that contains a view of that item's metadata, displayed as source code, appears in the code editor. The name of the type, followed by **[from metadata]**, appears on the document's tab.

For example, if you run the **Go To Definition** command for `Console`, metadata for `Console` appears in the code editor as C# source code. The code resembles its declaration, but does not show an implementation.

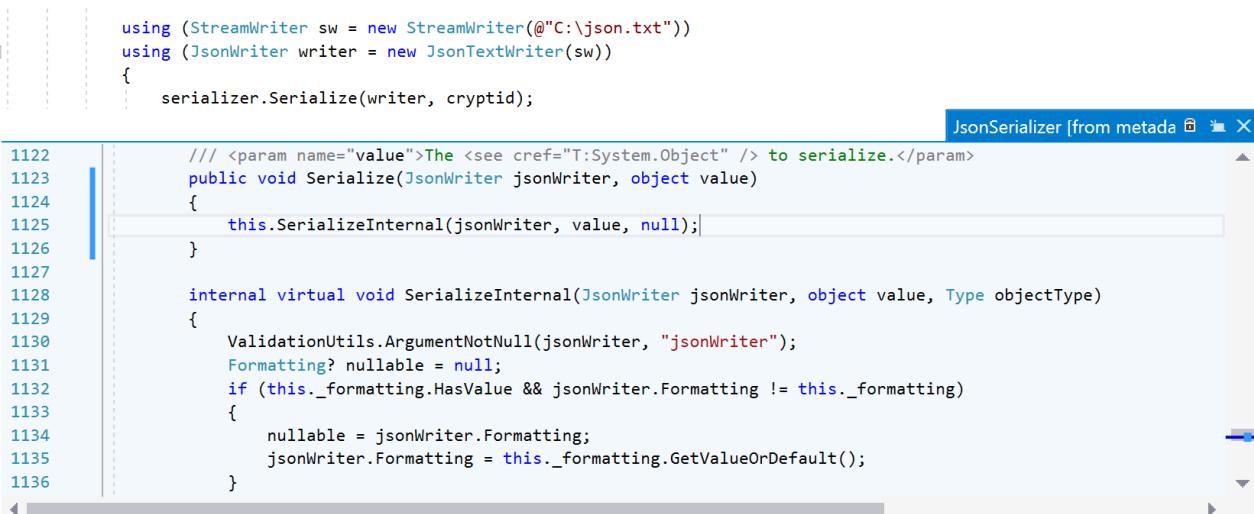


NOTE

When you try to run the **Go To Definition** or **Peek Definition** command for types or members that are marked as internal, Visual Studio does not display their metadata as source code, regardless of whether the referencing assembly is a friend or not.

View decompiled source definitions instead of metadata (C#)

You can set an option to see decompiled source code when you view the definition of a C# type or member whose source code is unavailable. To turn on this feature, choose **Tools > Options** from the menu bar. Then, expand **Text Editor > C# > Advanced**, and select **Enable navigation to decompiled sources**.



NOTE

Visual Studio reconstructs method bodies using ILSpy decompilation. The first time you access this feature, you must agree to a legal disclaimer regarding software licensing and copyright and trademark laws.

See also

- [Navigate code](#)
- [How to: View and edit code by using Peek Definition \(Alt+F12\)](#)

How to: View and edit code by using Peek Definition (Alt+F12)

10/18/2019 • 3 minutes to read • [Edit Online](#)

You can use the **Peek Definition** command to view and edit code without switching away from the code that you're writing. **Peek Definition** and **Go To Definition** show the same information, but **Peek Definition** shows it in a pop-up window, and **Go To Definition** shows the code in a separate code window. **Go To Definition** causes your context (that is, the active code window, current line, and cursor position) to switch to the definition code window. By using **Peek Definition**, you can view and edit the definition and move around inside the definition file while keeping your place in the original code file.

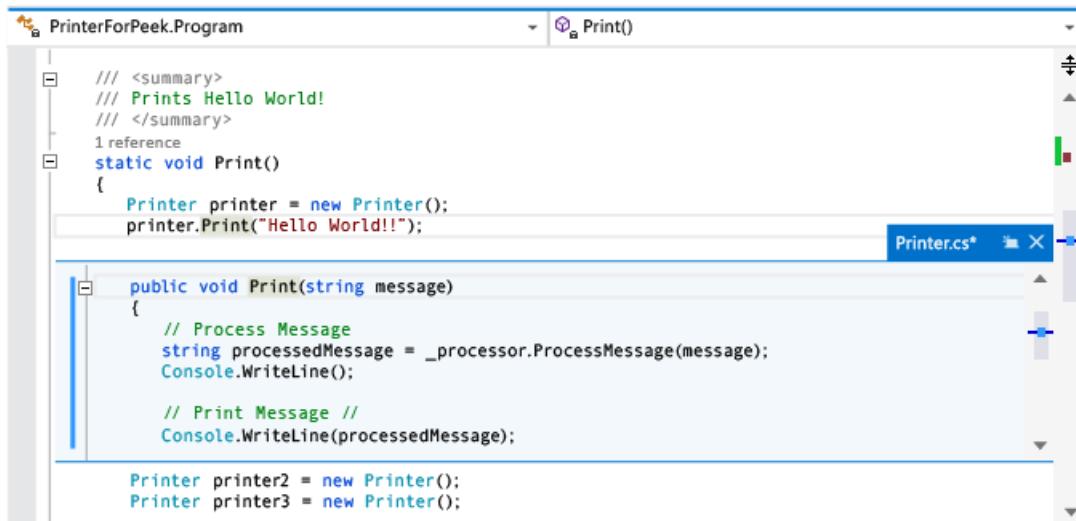
You can use **Peek Definition** with C#, Visual Basic, and C++ code. In Visual Basic, **Peek Definition** shows a link to the **Object Browser** for symbols that don't have definition metadata (for example, .NET types that are built in).

Use Peek Definition

Open a Peek Definition window

1. You can peek a definition by choosing **Peek Definition** from the right-click menu for a type or member that you want to explore. If the option is enabled, you can also peek a definition using the mouse, by pressing **Ctrl** (or another modifier) and clicking the member name. Or, from the keyboard, press **Alt+F12**.

This illustration shows the **Peek Definition** window for a method that's named `Print()`:

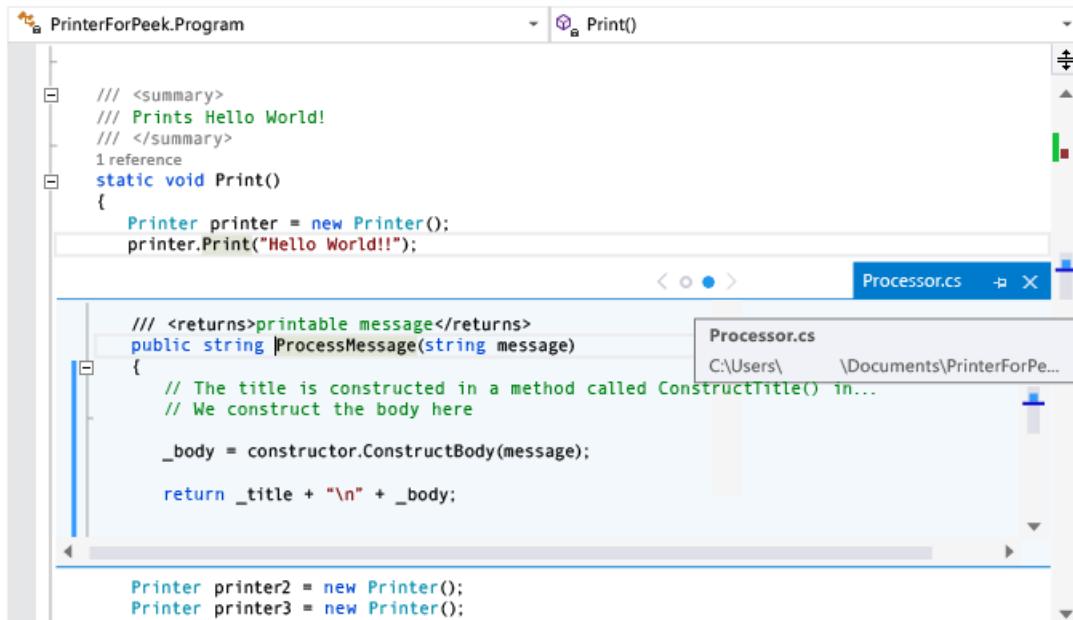


The definition window appears below the `printer.Print("Hello World!")` line in the original file. The window doesn't hide any of the code in your original file. The lines that follow `printer.Print("Hello World!")` appear under the definition window.

2. You can move the cursor to different locations in the peek definition window. You can also still move around in the original code window.
3. You can copy a string from the definition window and paste it in the original code. You can also drag and drop the string from the definition window to the original code without deleting it from the definition window.
4. You can close the definition window by choosing the **Esc** key or the **Close** button on the definition window tab.

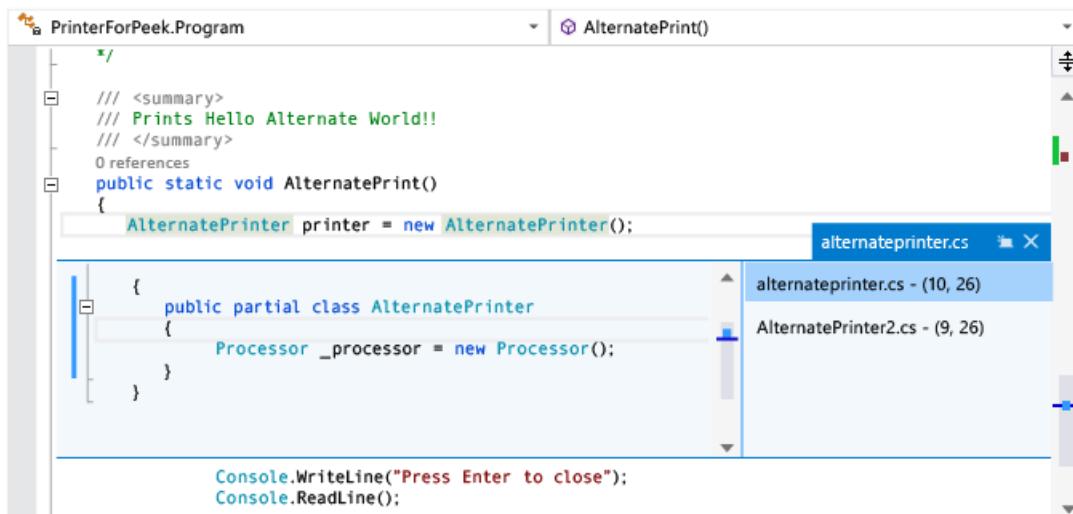
Open a Peek Definition window from within a Peek Definition window

If you already have a **Peek Definition** window open, you can call **Peek Definition** again on the code in that window. Another definition window opens. A set of breadcrumb dots appears next to the definition window tab, which you can use to navigate between definition windows. The tooltip on each dot shows the file name and path of the definition file that the dot represents.



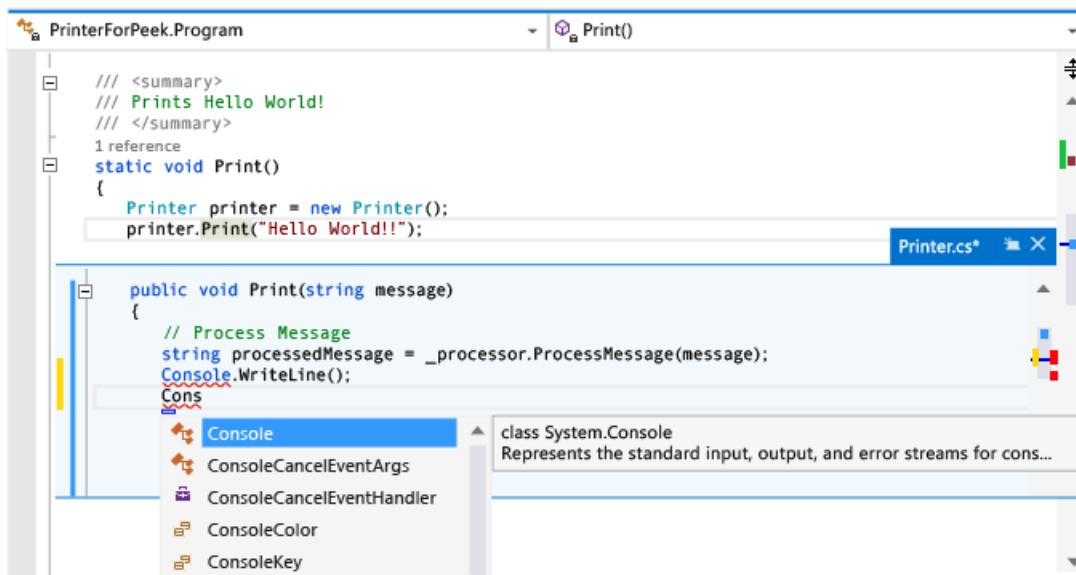
Peek Definition with multiple results

If you use **Peek Definition** on code that has more than one definition (for example, a partial class), a result list appears to the right of the code definition view. You can choose any result in the list to display its definition.



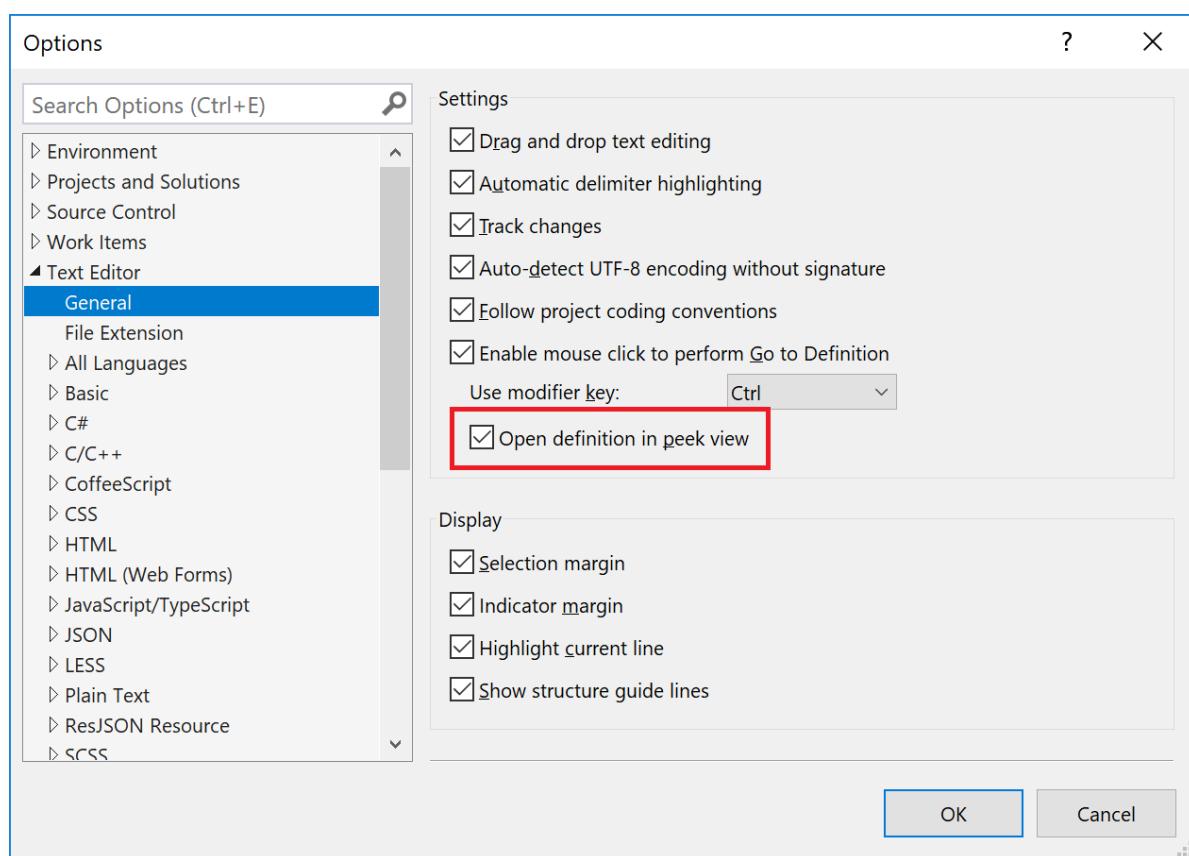
Edit inside the Peek Definition window

When you start to edit inside a **Peek Definition** window, the file that you're modifying automatically opens as a separate tab in the code editor and reflects the changes that you've made. You can continue to make, undo, and save changes in the **Peek Definition** window, and the tab will continue to reflect those changes. Even if you close the **Peek Definition** window without saving your changes, you can make, undo, and save more changes in the tab, picking up exactly where you left off in the **Peek Definition** window.



To change options for Peek Definition

1. Go to **Tools > Options > Text Editor > General**.
2. Select the option **Open definition in peek view**.
3. Click **OK** to close the **Options** dialog box.



Keyboard shortcuts for Peek Definition

You can use these keyboard shortcuts with the **Peek Definition** window:

FUNCTIONALITY	KEYBOARD SHORTCUT
Open the definition window	Alt+F12
Close the definition window	Esc

FUNCTIONALITY	KEYBOARD SHORTCUT
Promote the definition window to a regular document tab	Shift+Alt+Home
Navigate between definition windows	Ctrl+Alt+- and Ctrl+Alt+=
Navigate between multiple results	F8 and Shift+F8
Toggle between the code editor window and the definition window	Shift+Esc

NOTE

You can also use the same keyboard shortcuts to edit code in a **Peek Definition** window as you use elsewhere in Visual Studio.

See also

- [Navigate code](#)
- [Go To Definition and Peek Definition](#)
- [Productivity features in Visual Studio](#)

Find code using Go To commands

10/18/2019 • 2 minutes to read • [Edit Online](#)

Visual Studio's **Go To** commands perform a focused search of your code to help you quickly find specified items. You can go to a specific line, type, symbol, file, and member from a simple, unified interface.

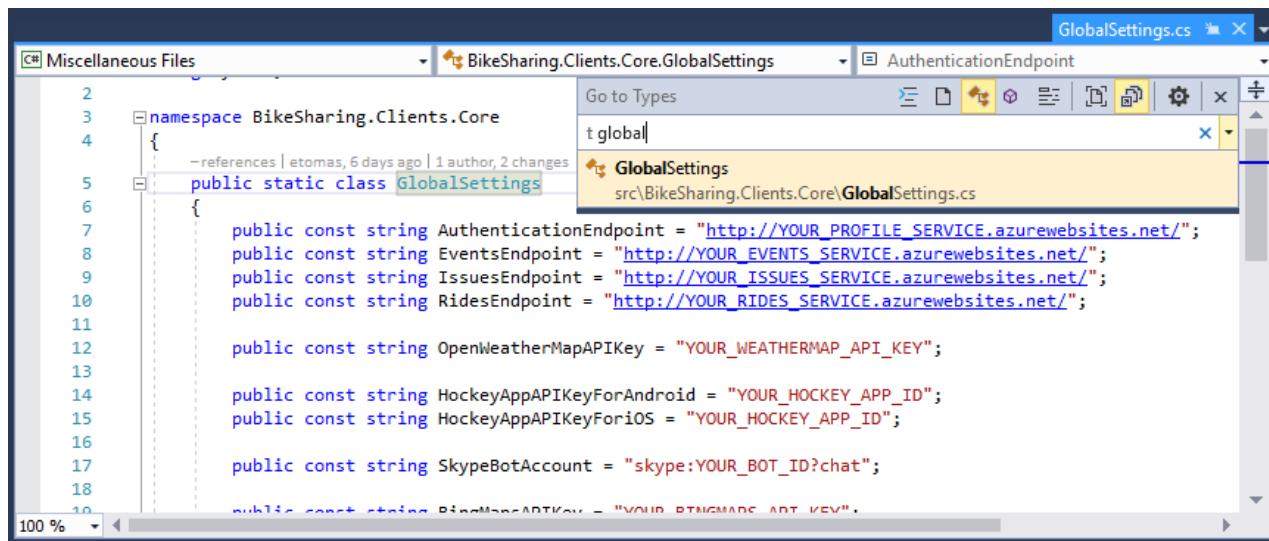
How to use it

INPUT	FUNCTION
Keyboard	Press Ctrl+T or Ctrl+,
Mouse	Select Edit > Go To > Go To All

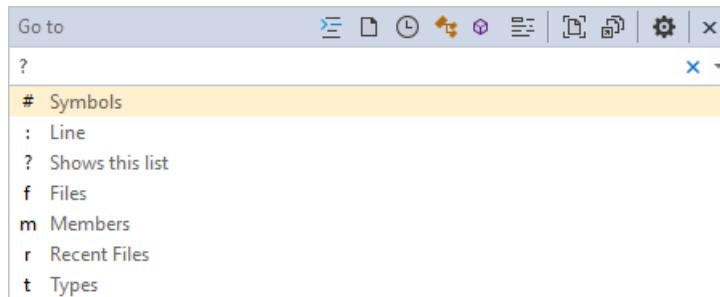
A small window is displayed at the top right of your code editor.



As you type in the text box, the results appear in a drop-down list below the text box. To go to an element, choose it in the list.



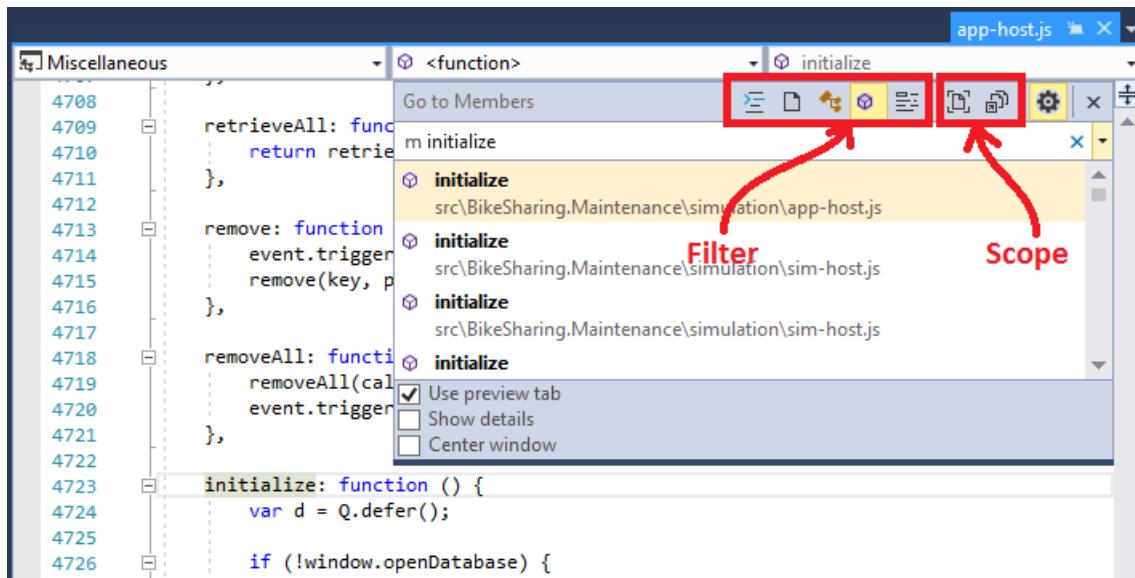
You can also enter a question mark (?) to get additional help.



Filtered searches

By default, the specified item is searched for in all solution items. However, you can limit your code search to

specific element types by prefacing the search terms with certain characters. You can also quickly change the search filter by choosing buttons on the **Go To** dialog box toolbar. Buttons that change the type filters are on the left side, and buttons that change the scope of the search are on the right side.



Filter to a specific type of code element

To narrow your search to a specific type of code element, you can either specify a prefix in the search box, or select one of the five filter icons:

PREFIX	ICON	SHORTCUT	DESCRIPTION
:		Ctrl+G	Go to the specified line number
f		Ctrl+1, Ctrl+F	Go to the specified file
r		Ctrl+1, Ctrl+R	Go to the specified, recently visited file
t		Ctrl+1, Ctrl+T	Go to the specified type
m		Ctrl+1, Ctrl+M	Go to the specified member
#		Ctrl+1, Ctrl+S	Go to the specified symbol

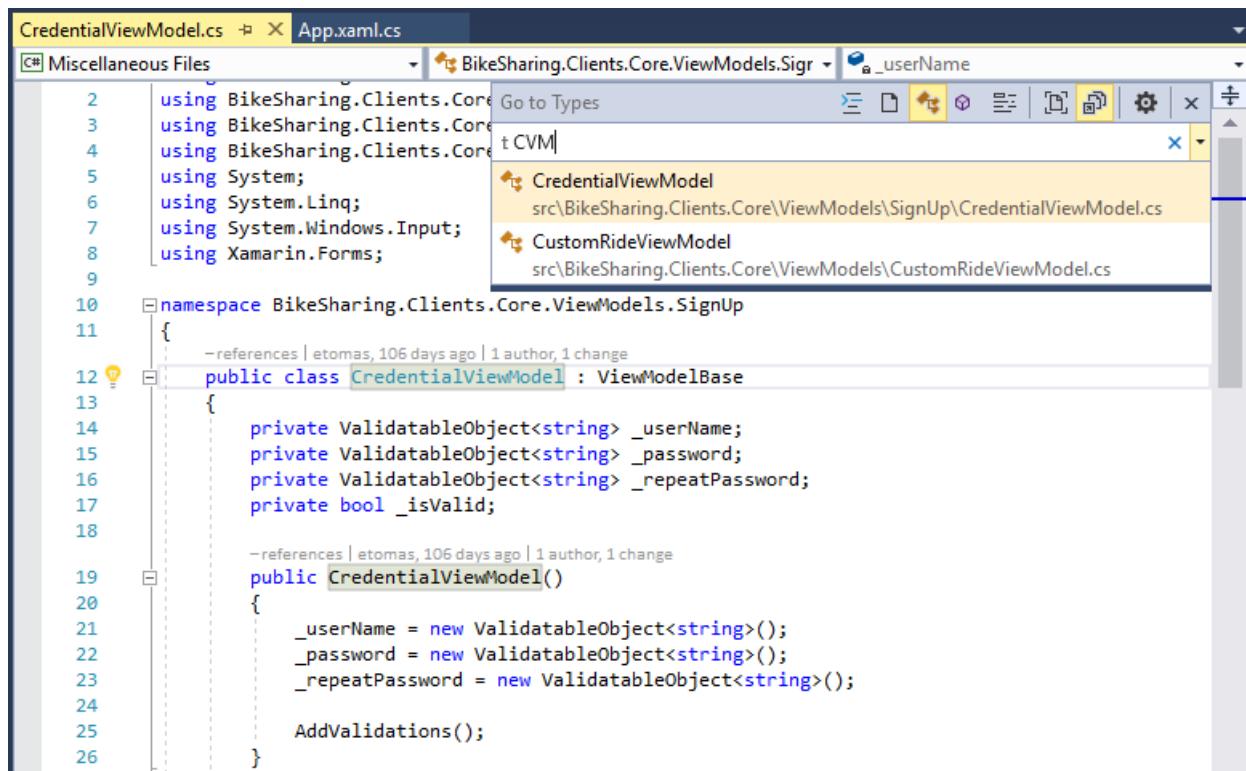
Filter to a specific location

To narrow your search to a specific location, select one of the two document icons:

ICON	DESCRIPTION
	Search current document only
	Search external documents in addition to those located in the project/solution

Camel casing

If you use [camel casing](#) in your code, you can find code elements faster by entering only the capital letters of the code element name. For example, if your code has a type called `CredentialViewModel`, you can narrow down the search by choosing the **Type** filter (**t**) and then entering just the capital letters of the name (`cvm`) in the Go To dialog box. This feature can be helpful if your code has long names.



Settings

Selecting the gear icon lets you change how this feature works:

SETTING	DESCRIPTION
Use preview tab	Display the selected item immediately in the IDE's preview tab
Show details	Display project, file, line, and summary information from documentation comments in the window
Center window	Move this window to the top-center of the code editor, instead of the top-right

See also

- [Navigate code](#)
- [Go To Line dialog box](#)
- [Go To Definition and Peek Definition](#)

How to: Change text case in the editor

10/18/2019 • 2 minutes to read • [Edit Online](#)

You can use menu commands or keyboard shortcuts to convert the case of selected text to all upper case or to all lower case.

NOTE

The dialog boxes and menu commands you see might differ from those described in this article, which are based on the **General** environment settings. To change your environment settings, choose **Tools** > **Import and Export Settings**, and then choose **Reset all settings**.

To change text case

1. Select the text you want to convert.
2. To convert text to all upper case, choose **Edit** > **Advanced** > **Make Uppercase** or press **Ctrl+Shift+U**.

To convert text to all lower case, choose **Edit** > **Advanced** > **Make Lowercase** or press **Ctrl+U**. (If you have the C++ development workload installed, this keybinding may be used by a different command.)

TIP

To revert to the previous case formatting before this change, select **Undo** from the **Edit** menu.

See also

- [Features of the code editor](#)

How to: Manage editor modes

10/18/2019 • 2 minutes to read • [Edit Online](#)

You can display the Visual Studio code editor in various display modes.

NOTE

The dialog boxes and menu commands you see might differ from those described in this article depending on your active settings or edition. To change your settings, for example to **General** or **Visual C++** settings, choose **Tools > Import and Export Settings**, and then choose **Reset all settings**.

Enable full screen mode

You can choose to hide all tool windows and view only document windows by enabling **Full Screen** mode.

- Press **Alt+Shift+Enter** to enter or exit **Full Screen** mode.
-- or --
- Issue the command `View.Fullscreen` in the **Command** window.

Enable virtual space mode

In **Virtual Space** mode, spaces are inserted at the end of each line of code. Select this option to position comments at a consistent point next to your code.

1. Select **Options** from the **Tools** menu.
2. Expand the **Text Editor** folder, and choose **All Languages** to set this option globally, or choose a specific language folder. For example, to turn on line numbers only in Visual Basic, choose the **Basic > Text Editor** node.
3. Select **General** options, and under **Settings**, select **Enable Virtual Space**.

NOTE

Virtual Space is enabled in **Column Selection** mode. When **Virtual Space** mode is not enabled, the insertion point moves from the end of one line directly to the first character of the next.

See also

- [Customize window layouts in Visual Studio](#)
- [Fonts and Colors, Environment, Options dialog box](#)

How to: Manage editor windows

10/18/2019 • 2 minutes to read • [Edit Online](#)

You can work on code in several locations at once. Do this by splitting an editor window, or by opening several instances of editor windows.

NOTE

Not all editor windows support multiple instances.

Split an editor window

An instance of an editor window can be split into two separate views for easier editing.

To split a pane

1. Click within the editor window to give it focus.
2. From the **Window** menu, select **Split**.

The editing area divides into two panes separated by a splitter bar. You can scroll these panes independently to view and edit different parts of the active document at the same time. Any changes made in one pane are reflected in the other.

TIP

To make one pane larger than the other, drag the splitter bar upward or downward.

To return to single-pane view

- From the **Window** menu, select **Remove Split**.

Create new windows

You can also create multiple instances of an editor window. This feature allows you to open a lengthy document in more than one instance of an editor, so that you can view and edit different sections simultaneously in separate, full-sized editor windows.

- On the **Window** menu, click **New Window**.

A new tabbed instance of the editor is added.

See also

- [Features of the code editor](#)
- [Customize window layouts](#)

How to: Change fonts and colors for the editor in Visual Studio

10/18/2019 • 2 minutes to read • [Edit Online](#)

You can change the default font face, adjust the font size, and change the foreground and background colors for various text **Display items** in the code editor. When you change font settings, keep the following information in mind:

- The settings for **Font** and **Size** are global for all text elements in all Visual Studio editors.
- The names of fixed width fonts are listed in bold.
- **Item foreground**, **Item background**, and **Bold** options can be set for each type of text element. For example, if you change colors and select **Bold** for **Comment** and **Bookmarks**, other types of text elements will be unaffected.

Change the default font face, size, and colors

1. Select **Options** from the **Tools** menu. Under **Environment**, select [Fonts and Colors](#).
2. In **Show settings for**, select **Text Editor**.
3. Modify the **Font** and **Size** options to change the font face and size for all text elements in all editors.
4. Select the appropriate item in **Display items**, and then modify the **Item foreground** and **Item background** options.

TIP

Click **Use Defaults** to reset the default settings.

5. Click **OK**.

See also

- [Features of the code editor](#)
- [How to: Change fonts and colors in Visual Studio](#)

How to: Manage word wrap in the editor

10/18/2019 • 2 minutes to read • [Edit Online](#)

You can set and clear the **Word wrap** option. When this option is set, the portion of a long line that extends beyond the current width of the Code Editor window is displayed on the next line. When this option is cleared, for example, to facilitate the use of line numbering, you can scroll to the right to see the ends of long lines.

NOTE

This topic applies only to Visual Studio on Windows. Visual Studio for Mac does not currently support word wrap.

To set word wrap preferences

1. On the **Tools** menu, select **Options**.
2. In the **Text Editor** folder, choose the **General** options in the **All Languages** subfolder to set this option globally.
— or —

Choose the **General** options in the subfolder for the language in which you are programming.

3. Under **Settings**, select or clear the **Word wrap** option.

When the **Word wrap** option is selected, the **Show visual glyphs for word wrap** option is enabled.

4. Select the **Show visual glyphs for Word Wrap** option if you prefer to display a return-arrow indicator where a long line wraps onto a second line. Clear this option if you prefer not to display indicator arrows.

NOTE

These reminder arrows are not added to your code; they are for display purposes only.

Known issues

If you're familiar with word wrap in Notepad++, Sublime Text, or Visual Studio Code, be aware of the following issues where Visual Studio behaves differently to other editors:

- [Triple click doesn't select whole line](#)
- [Cut command doesn't delete whole line](#)
- [Pressing End key twice does not move cursor to end of line](#)

See also

- [Features of the code editor](#)

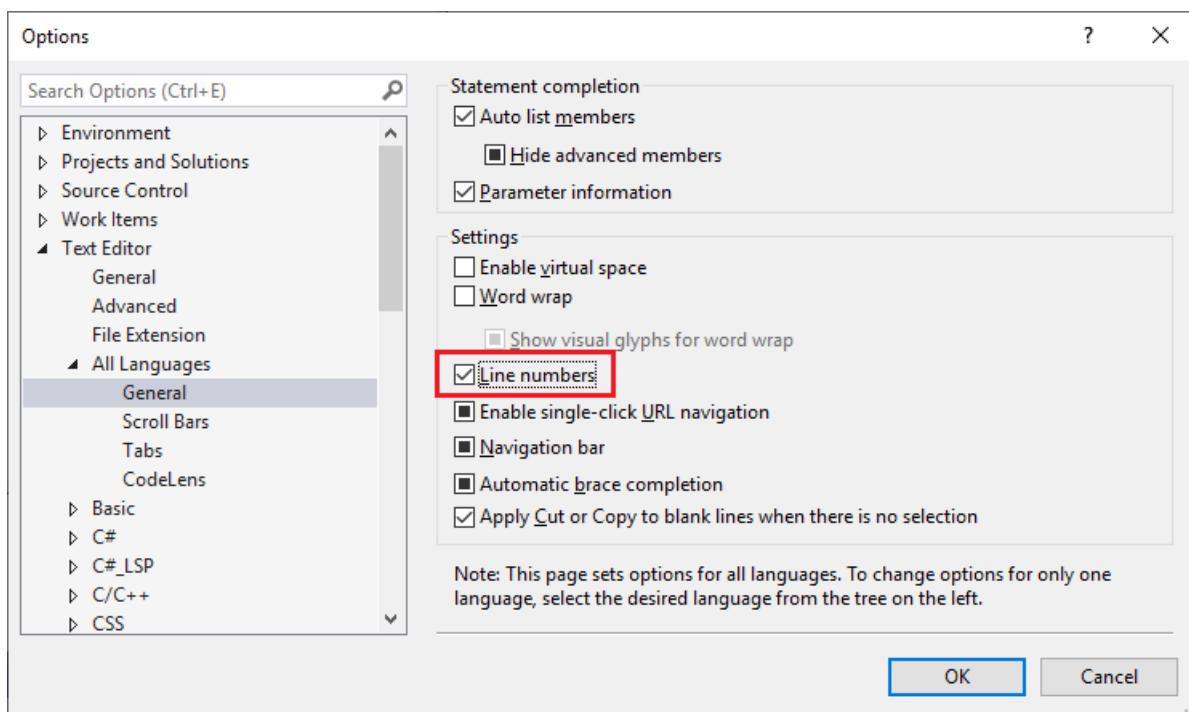
How to: Display line numbers in the editor

10/18/2019 • 2 minutes to read • [Edit Online](#)

You can display or hide line numbering in your code.

Display line numbers in code

1. On the menu bar, choose **Tools > Options**. Expand the **Text Editor** node, and then select either the language you're using or **All Languages** to turn on line numbers in all languages. (Or, type **line number** in the search box and choose **Turn line numbers on or off** from the results.)
2. Select the **Line numbers** checkbox.



TIP

Line numbers aren't added to your code; they're just for reference. If you want line numbers to print, in the **Print** dialog box, select the **Include line numbers** check box.

See also

- [Features of the code editor](#)

How to: Display URLs as Links in the Editor

10/18/2019 • 2 minutes to read • [Edit Online](#)

You can choose to have the Code Editor treat Uniform Resource Locators (URLs) in your code as active links. When you use this feature, URLs:

- Appear underlined.
- Display a **ToolTip** when you hover over them.
- Attempt to open the website indicated when you **Ctrl+click** on the link. By default, the website is displayed in the internal web browser.

Display URLs as links

1. On the **Tools** menu, click **Options**.
2. Click **Text Editor**.
3. To change the option for only one language, expand the folder for that language and choose **General**.
—or—
To change the option for all languages, expand the **All Languages** folder and choose **General**.
4. Under **Display**, select **Enable single-click URL navigation**.

See also

- [Features of the code editor](#)

Set language-specific editor options

10/18/2019 • 2 minutes to read • [Edit Online](#)

Visual Studio offers a variety of text editor options that apply to specific programming languages. You can configure options in the **Options** dialog box, which is accessed from the **Tools** menu. You can also configure some editor settings on a project- or codebase-basis by [creating an EditorConfig file](#).

Settings available in the Options dialog box

[Options, Text Editor, Basic \(Visual Basic\)](#)

Describes settings for end constructs, code reformatting, outlining, and error correction suggestions, among others, for Visual Basic code.

[Options, Text Editor, C/C++, Formatting](#)

Describes outlining, indenting, Quick Info, and other settings for C and C++ code.

[Options, Text Editor, C/C++, Advanced](#)

Describes settings for IntelliSense and database files when using C and C++.

[Options, Text Editor, C#, Formatting](#)

Describes settings for indenting, new line formatting, and wrapping text, among others, for C#.

[Options, Text Editor, C#, Advanced](#)

Describes outlining, error identification, and XML documentation comment settings for C#.

[Options, Text Editor, C#, IntelliSense](#)

Describes settings that specify how the IntelliSense completion list behaves when you work in C# code.

[Options, Text Editor, XAML, Formatting](#)

Describes settings for element and attribute arrangement in XAML documents.

See also

- [Customize the editor](#)
- [Create portable, custom editor settings with EditorConfig](#)
- [Personalize the Visual Studio IDE](#)

Create portable, custom editor settings with EditorConfig

10/25/2019 • 8 minutes to read • [Edit Online](#)

You can add an [EditorConfig](#) file to your project or codebase to enforce consistent coding styles for everyone that works in the codebase. EditorConfig settings take precedence over global Visual Studio text editor settings. This means that you can tailor each codebase to use text editor settings that are specific to that project. You can still set your own personal editor preferences in the Visual Studio **Options** dialog box. Those settings apply whenever you're working in a codebase without an *.editorconfig* file, or when the *.editorconfig* file doesn't override a particular setting. An example of such a preference is indent style—tabs or spaces.

EditorConfig settings are supported by numerous code editors and IDEs, including Visual Studio. It's a portable component that travels with your code, and can enforce coding styles even outside of Visual Studio.

When you add an EditorConfig file to your project in Visual Studio, new lines of code are formatted according to the EditorConfig settings. The formatting of existing code is not changed unless you run one of the following commands:

- [Code Cleanup \(Ctrl+K, Ctrl+E\)](#), which applies any white space settings, such as indent style, and selected code style settings, such as how to sort `using` directives.
- [Edit > Advanced > Format Document](#) (or [Ctrl+K, Ctrl+D](#) in the default profile), which only applies white space settings, such as indent style.

When you add an EditorConfig file to your project in Visual Studio, new lines of code are formatted according to the EditorConfig settings. The formatting of existing code is not changed unless you run unless you format the document ([Edit > Advanced > Format Document](#) or [Ctrl+K, Ctrl+D](#) in the default profile). Formatting the document only affects white space settings, such as indent style, unless you've configured Format Document to [perform additional code cleanup](#).

You can define which EditorConfig settings you want **Format Document** to apply on the [Formatting options page](#).

NOTE

This topic applies to Visual Studio on Windows. For Visual Studio for Mac, see [EditorConfig in Visual Studio for Mac](#).

Code consistency

Settings in EditorConfig files enable you to maintain consistent coding styles and settings in a codebase, such as indent style, tab width, end of line characters, encoding, and more, regardless of the editor or IDE you use. For example, when coding in C#, if your codebase has a convention to prefer that indents always consist of five space characters, documents use UTF-8 encoding, and each line always ends with a CR/LF, you can configure an *.editorconfig* file to do that.

Coding conventions you use on your personal projects may differ from those used on your team's projects. For example, you might prefer that when you're coding, indenting adds a tab character. However, your team might prefer that indenting adds four space characters instead of a tab character. EditorConfig files resolve this problem by enabling you to have a configuration for each scenario.

Because the settings are contained in a file in the codebase, they travel along with that codebase. As long as you

open the code file in an EditorConfig-compliant editor, the text editor settings are implemented. For more information about EditorConfig files, see the [EditorConfig.org](#) website.

NOTE

Conventions that are set in an EditorConfig file cannot currently be enforced in a CI/CD pipeline as build errors or warnings. Any style deviations appear only in the Visual Studio editor and **Error List**.

Supported settings

The editor in Visual Studio supports the core set of [EditorConfig properties](#):

- indent_style
- indent_size
- tab_width
- end_of_line
- charset
- trim_trailing whitespace
- insert_final_newline
- root

EditorConfig editor settings are supported in all Visual Studio-supported languages except for XML. In addition, EditorConfig supports [code style](#) conventions including [language](#), [formatting](#), and [naming](#) conventions for C# and Visual Basic.

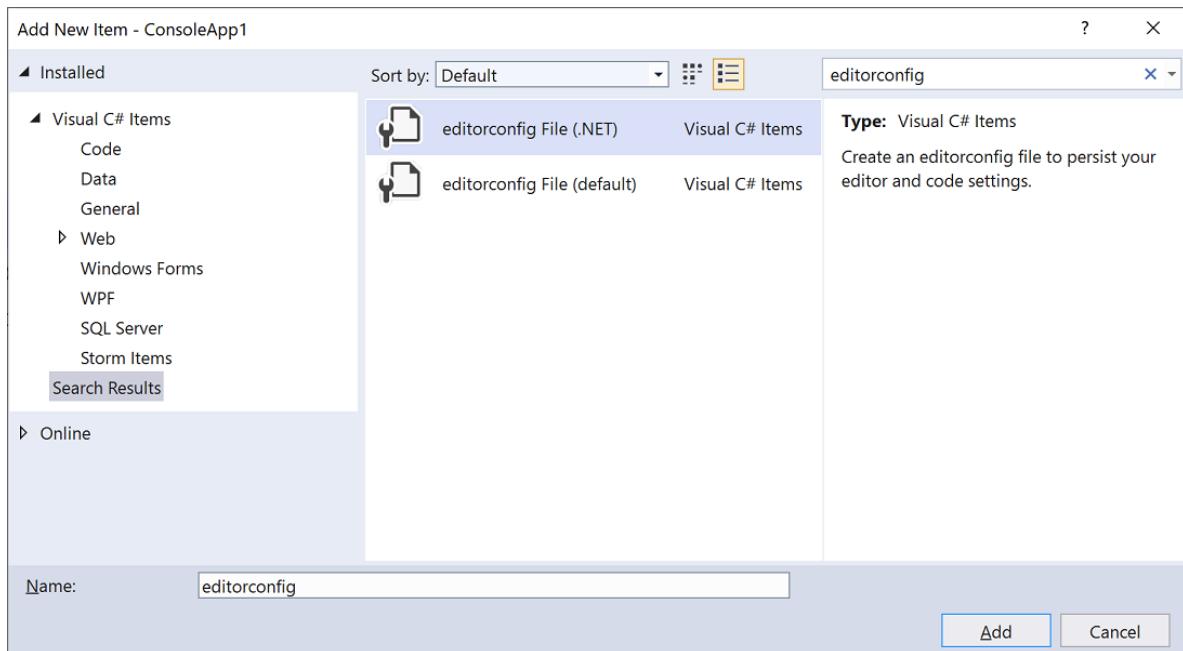
Add and remove EditorConfig files

When you add an EditorConfig file to your project or codebase, any new lines of code you write are formatted according to the EditorConfig file. However, adding an EditorConfig file does not convert existing styles to the new ones until you format the document or run [Code Cleanup](#). For example, if you have indents in your file that are formatted with tabs and you add an EditorConfig file that indents with spaces, the indent characters are not automatically converted to spaces. When you format the document (**Edit > Advanced > Format Document** or **Ctrl+K, Ctrl+D**), the white space settings in the EditorConfig file are applied to existing lines of code.

If you remove an EditorConfig file from your project or codebase and you want new lines of code to be formatted according to the global editor settings, you must close and reopen any open code files.

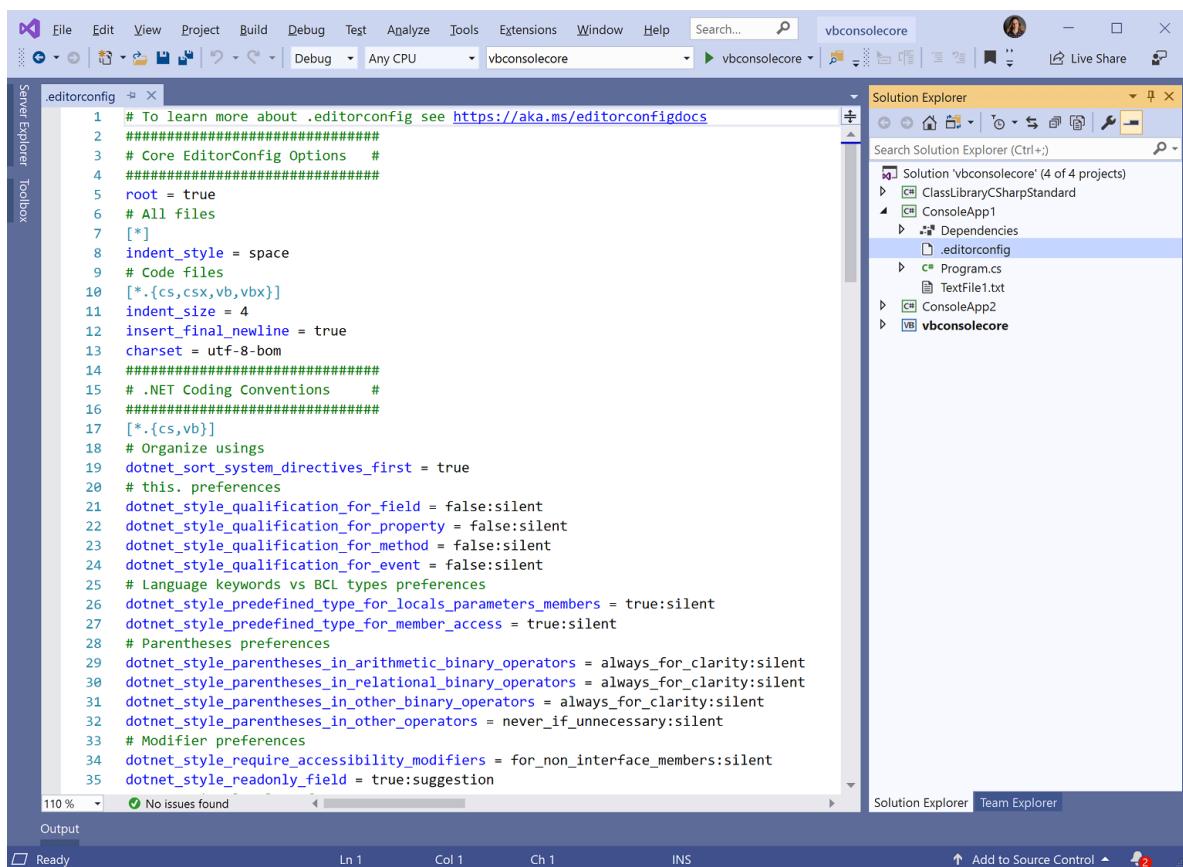
Add an EditorConfig file to a project

1. Open a project or solution in Visual Studio. Select either the project or solution node, depending on whether your *.editorconfig* settings should apply to all projects in the solution or just one. You can also select a folder in your project or solution to add the *.editorconfig* file to.
2. From the menu bar, choose **Project > Add New Item**, or press **Ctrl+Shift+A**.
The **Add New Item** dialog box opens.
3. In the search box, search for **editorconfig**.
Two **editorconfig File** item templates are shown in the search results.



4. Select the **editorconfig File (default)** template to add an EditorConfig file prepopulated with two core EditorConfig options for indent style and size. Or, select the **editorconfig File (.NET)** template to add an EditorConfig file prepopulated with default .NET code style, formatting, and naming conventions.

An *.editorconfig* file appears in Solution Explorer, and it opens in the editor.



5. Edit the file as desired.

Other ways to add an EditorConfig file

There are a couple other ways you can add an EditorConfig file to your project:

- The [code inference feature](#) of IntelliCode for Visual Studio infers your code styles from existing code. It then creates a non-empty EditorConfig file with your code-style preferences already defined.
- Starting in Visual Studio 2019, you can [generate an EditorConfig file based on your code-style settings](#) in

File hierarchy and precedence

When you add an `.editorconfig` file to a folder in your file hierarchy, its settings apply to all applicable files at that level and below. You can also override EditorConfig settings for a particular project, codebase, or part of a codebase, such that it uses different conventions than other parts of the codebase. This can be useful when you incorporate code from somewhere else, and don't want to change its conventions.

To override some or all of the EditorConfig settings, add an `.editorconfig` file at the level of the file hierarchy you want those overridden settings to apply. The new EditorConfig file settings apply to files at the same level and any subdirectories.

```
.editorconfig
file1
file2
└── .editorconfig
    ├── file3
    └── file4
```

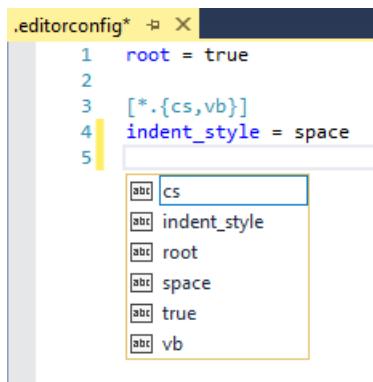
If you want to override some but not all of the settings, specify just those settings in the `.editorconfig` file. Only those properties that you explicitly list in the lower-level file are overridden. Other settings from higher-level `.editorconfig` files continue to apply. If you want to ensure that *no* settings from *any* higher-level `.editorconfig` files are applied to this part of the codebase, add the `root=true` property to the lower-level `.editorconfig` file:

```
# top-most EditorConfig file
root = true
```

EditorConfig files are read top to bottom. If there are multiple properties with the same name, the most recently found property with that name takes precedence.

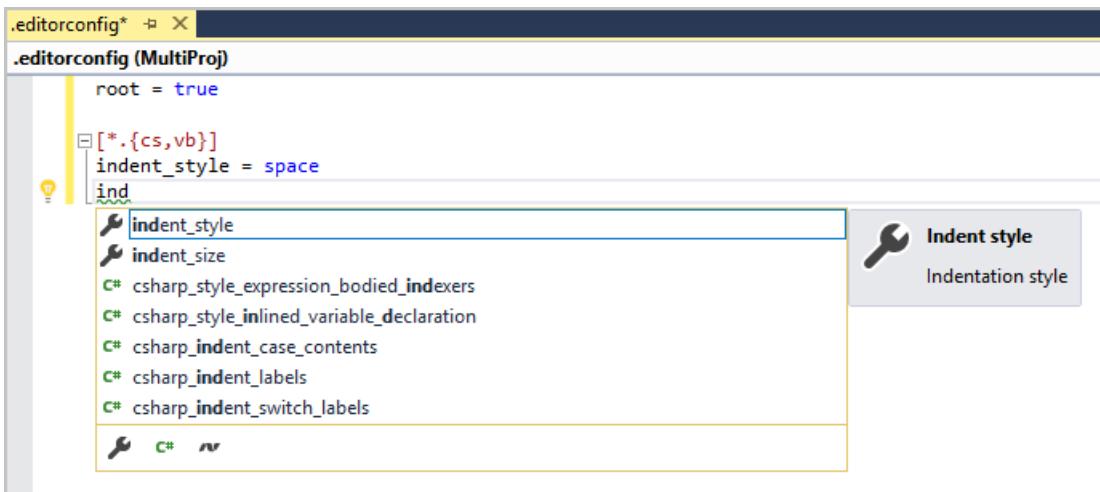
Edit EditorConfig files

Visual Studio helps you edit `.editorconfig` files by providing IntelliSense completion lists.



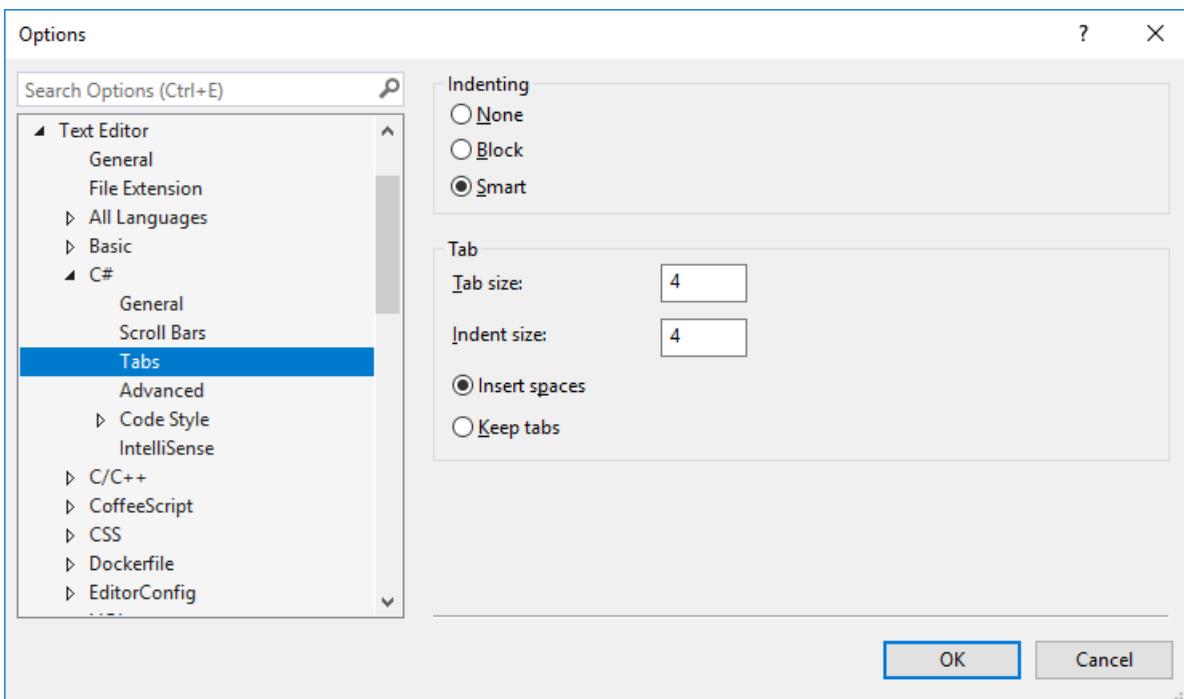
After you've edited your EditorConfig file, you must reload your code files for the new settings to take effect.

If you edit numerous `.editorconfig` files, you may find the [EditorConfig Language Service extension](#) helpful. Some of the features of this extension include syntax highlighting, improved IntelliSense, validation, and code formatting.

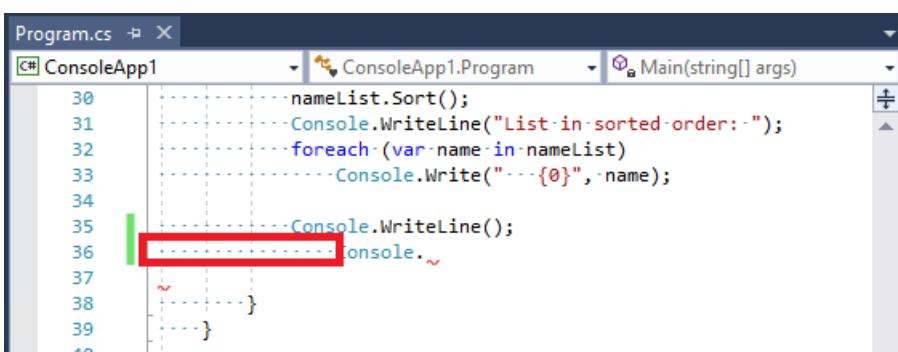


Example

The following example shows the indent state of a C# code snippet before and after adding an `.editorconfig` file to the project. The **Tabs** setting in the **Options** dialog box for the Visual Studio text editor is set to produce space characters when you press the **Tab** key.



As expected, pressing the **Tab** key on the next line indents the line by adding four additional white-space characters.

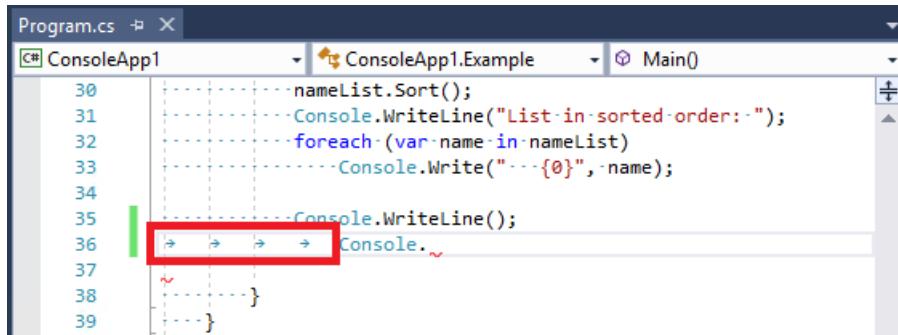


Add a new file called `.editorconfig` to the project, with the following contents. The `[*.cs]` setting means that this change applies only to C# code files in the project.

```
# Top-most EditorConfig file
root = true

# Tab indentation
[*.cs]
indent_style = tab
```

Now, when you press the **Tab** key, you get tab characters instead of spaces.

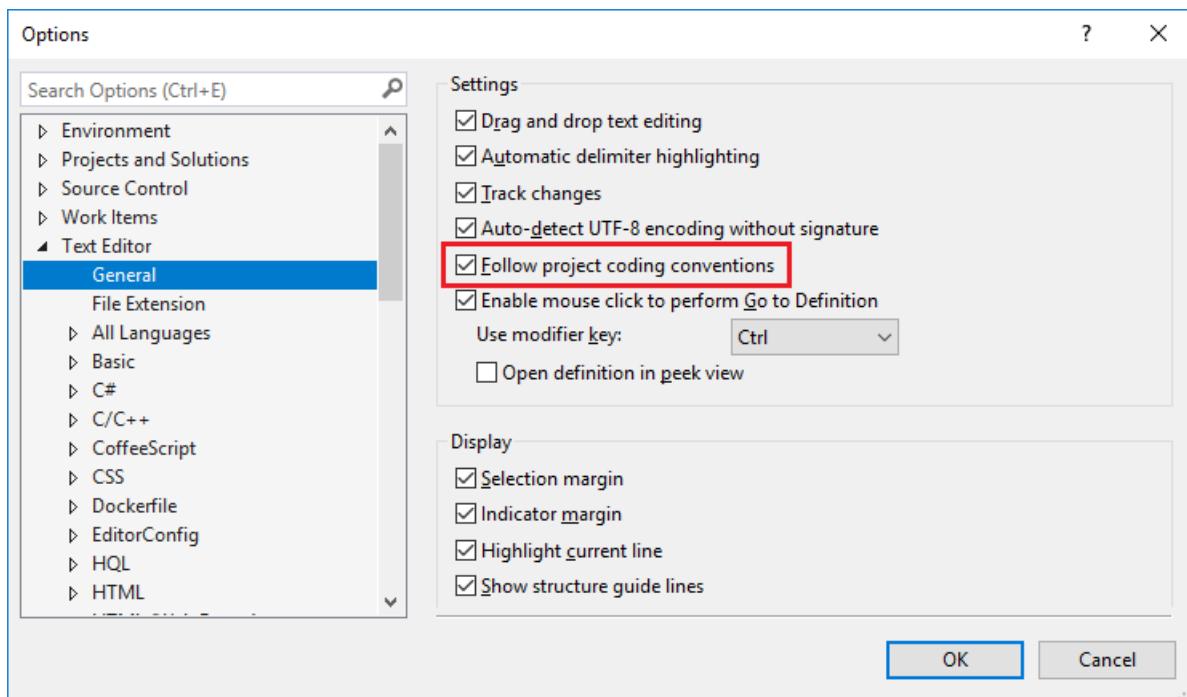


Troubleshoot EditorConfig settings

If there is an `EditorConfig` file anywhere in the directory structure at or above your project's location, Visual Studio applies the editor settings in that file to your editor. In this case, you may see the following message in the status bar:

"User preferences for this file type are overridden by this project's coding conventions."

This means that if any editor settings in **Tools > Options > Text Editor** (such as indent size and style, tab size, or coding conventions) are specified in an `EditorConfig` file at or above the project in the directory structure, the conventions in the `EditorConfig` file override the settings in **Options**. You can control this behavior by toggling the **Follow project coding conventions** option in **Tools > Options > Text Editor**. Unchecking the option turns off `EditorConfig` support for Visual Studio.



You can find any `.editorconfig` files in parent directories by opening a command prompt and running the following command from the root of the disk that contains your project:

```
dir .editorconfig /s
```

You can control the scope of your EditorConfig conventions by setting the `root=true` property in the `.editorconfig` file at the root of your repo or in the directory that your project resides. Visual Studio looks for a file named `.editorconfig` in the directory of the opened file and in every parent directory. The search ends when it reaches the root filepath, or if an `.editorconfig` file with `root=true` is found.

See also

- [.NET code style conventions](#)
- [Supporting EditorConfig for a language service](#)
- [EditorConfig.org](#)
- [Features of the code editor](#)
- [EditorConfig \(Visual Studio for Mac\)](#)

.NET coding convention settings for EditorConfig

10/18/2019 • 3 minutes to read • [Edit Online](#)

You can define and maintain consistent code style in your codebase with the use of an [EditorConfig](#) file.

EditorConfig includes several core formatting properties, such as `indent_style` and `indent_size`. In Visual Studio, .NET coding conventions settings can also be configured by using an EditorConfig file. You can enable or disable individual .NET coding conventions and configure the degree to which you want each rule enforced, via a severity level.

TIP

- When you define coding conventions in an EditorConfig file, you're configuring how you want the [code style analyzers](#) that are built into Visual Studio to analyze your code. The EditorConfig file is the configuration file for these analyzers.
- Code style preferences for Visual Studio can also be set in the [Text editor options](#) dialog. However, EditorConfig settings take precedence and preferences you set in [Options](#) aren't associated with a particular project.

Convention categories

There are three supported .NET coding convention categories:

- [Language conventions](#)

Rules pertaining to the C# or Visual Basic language. For example, you can specify rules around using `var` or explicit types when defining variables, or preferring expression-bodied members.

- [Formatting conventions](#)

Rules regarding the layout and structure of your code in order to make it easier to read. For example, you can specify rules around Allman braces, or preferring spaces in control blocks.

- [Naming conventions](#)

Rules regarding the naming of code elements. For example, you can specify that `async` methods must end in "Async".

Example EditorConfig file

To help you get started, here is an example `.editorconfig` file with the default options:

```
#####
# Core EditorConfig Options  #
#####

root = true

# All files
[*]
indent_style = space

# Code files
[*.cs,csx,vb,vbx]
indent_size = 4
insert_final_newline = true
charset = utf-8-bom
```

```

#####
# .NET Coding Conventions      #
#####

[*.cs,vb]
# Organize usings
dotnet_sort_system_directives_first = true
dotnet_separate_import_directive_groups = false

# this. preferences
dotnet_style_qualification_for_field = false:silent
dotnet_style_qualification_for_property = false:silent
dotnet_style_qualification_for_method = false:silent
dotnet_style_qualification_for_event = false:silent

# Language keywords vs BCL types preferences
dotnet_style_predefined_type_for_locals_parameters_members = true:silent
dotnet_style_predefined_type_for_member_access = true:silent

# Parentheses preferences
dotnet_style_parentheses_in_arithmetic_binary_operators = always_for_clarity:silent
dotnet_style_parentheses_in_relational_binary_operators = always_for_clarity:silent
dotnet_style_parentheses_in_other_binary_operators = always_for_clarity:silent
dotnet_style_parentheses_in_other_operators = never_if_unnecessary:silent

# Modifier preferences
dotnet_style_require_accessibility_modifiers = for_non_interface_members:silent
dotnet_style_READONLY_field = true:suggestion

# Expression-level preferences
dotnet_style_object_initializer = true:suggestion
dotnet_style_collection_initializer = true:suggestion
dotnet_style_explicit_tuple_names = true:suggestion
dotnet_style_null_propagation = true:suggestion
dotnet_style_coalesce_expression = true:suggestion
dotnet_style_prefer_is_null_check_over_reference_equality_method = true:silent
dotnet_style_prefer_inferred_tuple_names = true:suggestion
dotnet_style_prefer_inferred_anonymous_type_member_names = true:suggestion
dotnet_style_prefer_auto_properties = true:silent
dotnet_style_prefer_conditional_expression_over_assignment = true:silent
dotnet_style_prefer_conditional_expression_over_return = true:silent

#####
# Naming Conventions          #
#####

# Style Definitions
dotnet_naming_style.pascal_case_style.capitalization      = pascal_case

# Use PascalCase for constant fields
dotnet_naming_rule.constant_fields_should_be_pascal_case.severity = suggestion
dotnet_naming_rule.constant_fields_should_be_pascal_case.symbols   = constant_fields
dotnet_naming_rule.constant_fields_should_be_pascal_case.style    = pascal_case_style
dotnet_naming_symbols.constant_fields.applicable_kinds        = field
dotnet_naming_symbols.constant_fields.applicable_accessibilities = *
dotnet_naming_symbols.constant_fields.required_modifiers       = const

#####
# C# Code Style Rules         #
#####

[*.cs]
# var preferences
csharp_style_var_for_built_in_types = true:silent
csharp_style_var_when_type_is_apparent = true:silent
csharp_style_var_elsewhere = true:silent

# Expression-bodied members
csharp_style_expression_bodied_methods = false:silent

```

```
csharp_style_expression_bodied_methods = false:silent
csharp_style_expression_bodied_constructors = false:silent
csharp_style_expression_bodied_operators = false:silent
csharp_style_expression_bodied_properties = true:silent
csharp_style_expression_bodied_indexers = true:silent
csharp_style_expression_bodied_accessors = true:silent

# Pattern-matching preferences
csharp_style_pattern_matching_over_is_with_cast_check = true:suggestion
csharp_style_pattern_matching_over_as_with_null_check = true:suggestion

# Null-checking preferences
csharp_style_throw_expression = true:suggestion
csharp_style_conditional_delegate_call = true:suggestion

# Modifier preferences
csharp_preferred_modifier_order =
public,private,protected,internal,static,extern,new,virtual,abstract,sealed,override,readonly,unsafe,volatile
,async:suggestion

# Expression-level preferences
csharp_prefer_braces = true:silent
csharp_style_deconstructed_variable_declaration = true:suggestion
csharp_prefer_simple_default_expression = true:suggestion
csharp_style_pattern_local_over_anonymous_function = true:suggestion
csharp_style_inlined_variable_declaration = true:suggestion

#####
# C# Formatting Rules      #
#####

# New line preferences
csharp_new_line_before_open_brace = all
csharp_new_line_before_else = true
csharp_new_line_before_catch = true
csharp_new_line_before_finally = true
csharp_new_line_before_members_in_object_initializers = true
csharp_new_line_before_members_in_anonymous_types = true
csharp_new_line_between_query_expression_clauses = true

# Indentation preferences
csharp_indent_case_contents = true
csharp_indent_switch_labels = true
csharp_indent_labels = flush_left

# Space preferences
csharp_space_after_cast = false
csharp_space_after_keywords_in_control_flow_statements = true
csharp_space_between_method_call_parameter_list_parentheses = false
csharp_space_between_method_declaration_parameter_list_parentheses = false
csharp_space_between_parentheses = false
csharp_space_before_colon_in_inheritance_clause = true
csharp_space_after_colon_in_inheritance_clause = true
csharp_space_around_binary_operators = before_and_after
csharp_space_between_method_declaration_empty_parameter_list_parentheses = false
csharp_space_between_method_call_name_and_opening_parenthesis = false
csharp_space_between_method_call_empty_parameter_list_parentheses = false
csharp_space_after_comma = true
csharp_space_after_dot = false

# Wrapping preferences
csharp_preserve_single_line_statements = true
csharp_preserve_single_line_blocks = true

#####
# Visual Basic Code Style Rules  #
#####

[*.vb]
```

```
# Modifier preferences
visual_basic_preferred_modifier_order =
Partial,Default,Private,Protected,Public,Friend,NotOverridable,Overridable,MustOverride,Overloads,Overrides,
ustInherit,NotInheritable,Static,Shared,Shadows,ReadOnly,WriteOnly,Dim,Const,WithEvents,Widening,Narrowing,Cu
stom,Async:suggestion
```

See also

- [Quick Actions](#)
- [Create portable custom editor options](#)
- [.NET Compiler Platform's .editorconfig file](#)

Language conventions

10/18/2019 • 30 minutes to read • [Edit Online](#)

Language conventions for EditorConfig in Visual Studio fall into two categories: those that apply to Visual Basic and C#, and those that are C# specific. Language conventions affect how various aspects of a programming language are used, for example, modifiers and parentheses.

TIP

- To see the code examples in your preferred programming language, choose it using the language picker at the top-right corner of the browser window.



- Use the **In this article** links to jump to different sections of the page.

Rule format

Rules for language conventions have the following general format:

```
option_name = value:severity
```

For each language convention, you specify a value that defines if or when to prefer the style. Many rules accept a value of `true` (prefer this style) or `false` (do not prefer this style). Other rules accept values such as `when_on_single_line` or `never`. The second part of the rule specifies the `severity`.

NOTE

Because language conventions are enforced by analyzers, you can also set their severity by using the default configuration syntax for analyzers. The syntax takes the form `dotnet_diagnostic.<rule ID>.severity = <severity>`, for example, `dotnet_diagnostic.IDE0040.severity = silent`. For more information, see [Set rule severity in an EditorConfig file](#).

Severity levels

A language convention severity specifies the level at which to enforce that style. The following table lists the possible severity values and their effects:

SEVERITY	EFFECT
<code>error</code>	When this style rule is violated, show a compiler error.
<code>warning</code>	When this style rule is violated, show a compiler warning.
<code>suggestion</code>	When this style rule is violated, show it to the user as a suggestion. Suggestions appear as three gray dots under the first two characters.

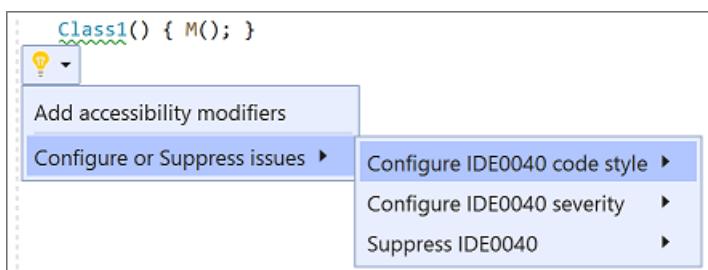
SEVERITY	EFFECT
<code>silent</code>	Do not show anything to the user when this rule is violated. Code generation features generate code in this style, however. Rules with <code>silent</code> severity participate in cleanup and appear in the Quick Actions and Refactorings menu.
<code>none</code>	Do not show anything to the user when this rule is violated. Code generation features generate code in this style, however. Rules with <code>none</code> severity never appear in the Quick Actions and Refactorings menu. In most cases, this is considered "disabled" or "ignored".

Automatically configure code styles

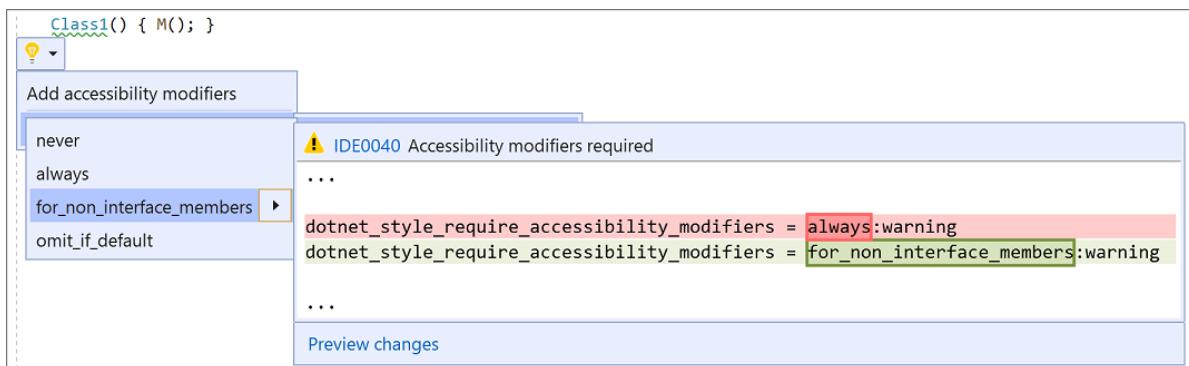
Starting in Visual Studio 2019 version 16.3, you can configure code style rules from the [Quick Actions](#) light bulb menu after a style violation occurs.

To change the code style convention:

1. Hover over the squiggle in the editor, and then open the light bulb menu that appears. Choose **Configure or Suppress issues > Configure <rule ID> code style**.



2. From there, choose one of the code style options.



Visual Studio adds or modifies the configuration setting in the EditorConfig file, as shown in the preview box.

To change the severity of the code style violation, follow the same steps, but choose **Configure <rule ID> severity** instead of **Configure <rule ID> code style**. For more information, see [Automatically configure rule severity](#).

.NET code style settings

The style rules in this section are applicable to both C# and Visual Basic.

- ["This." and "Me." qualifiers](#)
 - `dotnet_style_qualification_for_field`

- dotnet_style_qualification_for_property
- dotnet_style_qualification_for_method
- dotnet_style_qualification_for_event
- [Language keywords instead of framework type names for type references](#)
 - dotnet_style_predefined_type_for_locals_parameters_members
 - dotnet_style_predefined_type_for_member_access
- [Modifier preferences](#)
 - dotnet_style_require_accessibility_modifiers
 - csharp_preferred_modifier_order
 - visual_basic_preferred_modifier_order
 - dotnet_style_readonly_field
- [Parentheses preferences](#)
 - dotnet_style_parentheses_in_arithmetic_binary_operators
 - dotnet_style_parentheses_in_other_binary_operators
 - dotnet_style_parentheses_in_other_operators
 - dotnet_style_parentheses_in_relational_binary_operators
- [Expression-level preferences](#)
 - dotnet_style_object_initializer
 - dotnet_style_collection_initializer
 - dotnet_style_explicit_tuple_names
 - dotnet_style_prefer_inferred_tuple_names
 - dotnet_style_prefer_inferred_anonymous_type_member_names
 - dotnet_style_prefer_auto_properties
 - dotnet_style_prefer_is_null_check_over_reference_equality_method
 - dotnet_style_prefer_conditional_expression_over_assignment
 - dotnet_style_prefer_conditional_expression_over_return
 - dotnet_style_prefer_compound_assignment
- ["Null" checking preferences](#)
 - dotnet_style_coalesce_expression
 - dotnet_style_null_propagation

"This." and "Me." qualifiers

This style rule can be applied to fields, properties, methods, or events. A value of **true** means prefer the code symbol to be prefaced with `this.` in C# or `Me.` in Visual Basic. A value of **false** means prefer the code element *not* to be prefaced with `this.` or `Me.`.

These rules could appear in an `.editorconfig` file as follows:

```
# CSharp and Visual Basic code style settings:
[*.{cs,vb}]
dotnet_style_qualification_for_field = false:suggestion
dotnet_style_qualification_for_property = false:suggestion
dotnet_style_qualification_for_method = false:suggestion
dotnet_style_qualification_for_event = false:suggestion
```

`dotnet_style_qualification_for_field`

Rule name	dotnet_style_qualification_for_field

Rule ID	IDE0003 and IDE0009
Applicable languages	C# and Visual Basic
Values	<p><code>true</code> - Prefer fields to be prefaced with <code>this.</code> in C# or <code>Me.</code> in Visual Basic</p> <p><code>false</code> - Prefer fields <i>not</i> to be prefaced with <code>this.</code> or <code>Me.</code></p>
Visual Studio default	<code>false:silent</code>

Code examples:

```
// dotnet_style_qualification_for_field = true
this.capacity = 0;

// dotnet_style_qualification_for_field = false
capacity = 0;
```

```
' dotnet_style_qualification_for_field = true
Me.capacity = 0

' dotnet_style_qualification_for_field = false
capacity = 0
```

dotnet_style_qualification_for_property

Rule name	dotnet_style_qualification_for_property
Rule ID	IDE0003 and IDE0009
Applicable languages	C# and Visual Basic
Values	<p><code>true</code> - Prefer properties to be prefaced with <code>this.</code> in C# or <code>Me.</code> in Visual Basic</p> <p><code>false</code> - Prefer properties <i>not</i> to be prefaced with <code>this.</code> or <code>Me.</code></p>
Visual Studio default	<code>false:silent</code>

Code examples:

```
// dotnet_style_qualification_for_property = true
this.ID = 0;

// dotnet_style_qualification_for_property = false
ID = 0;
```

```
' dotnet_style_qualification_for_property = true
Me.ID = 0

' dotnet_style_qualification_for_property = false
ID = 0
```

dotnet_style_qualification_for_method

Rule name	dotnet_style_qualification_for_method
Rule ID	IDE0003 and IDE0009
Applicable languages	C# and Visual Basic
Values	<p><code>true</code> - Prefer methods to be prefaced with <code>this.</code> in C# or <code>Me..</code> in Visual Basic.</p> <p><code>false</code> - Prefer methods <i>not</i> to be prefaced with <code>this.</code> or <code>Me..</code>.</p>
Visual Studio default	<code>false:silent</code>

Code examples:

```
// dotnet_style_qualification_for_method = true
this.Display();

// dotnet_style_qualification_for_method = false
Display();
```

```
' dotnet_style_qualification_for_method = true
Me.Display()

' dotnet_style_qualification_for_method = false
Display()
```

dotnet_style_qualification_for_event

Rule name	dotnet_style_qualification_for_event
Rule ID	IDE0003 and IDE0009
Applicable languages	C# and Visual Basic
Values	<p><code>true</code> - Prefer events to be prefaced with <code>this.</code> in C# or <code>Me..</code> in Visual Basic.</p> <p><code>false</code> - Prefer events <i>not</i> to be prefaced with <code>this.</code> or <code>Me..</code>.</p>
Visual Studio default	<code>false:silent</code>

Code examples:

```
// dotnet_style_qualification_for_event = true
this.Elapsed += Handler;

// dotnet_style_qualification_for_event = false
Elapsed += Handler;
```

```
' dotnet_style_qualification_for_event = true
AddHandler Me.Elapsed, AddressOf Handler

' dotnet_style_qualification_for_event = false
AddHandler Elapsed, AddressOf Handler
```

Language keywords instead of framework type names for type references

This style rule can be applied to local variables, method parameters, and class members, or as a separate rule to type member access expressions. A value of **true** means prefer the language keyword (for example, `int` or `Integer`) instead of the type name (for example, `Int32`) for types that have a keyword to represent them. A value of **false** means prefer the type name instead of the language keyword.

These rules could appear in an `.editorconfig` file as follows:

```
# CSharp and Visual Basic code style settings:
[*.{cs,vb}]
dotnet_style_predefined_type_for_locals_parameters_members = true:suggestion
dotnet_style_predefined_type_for_member_access = true:suggestion
```

dotnet_style_predefined_type_for_locals_parameters_members

Rule name	dotnet_style_predefined_type_for_locals_parameters_members
Rule ID	IDE0012 and IDE0014
Applicable languages	C# and Visual Basic
Values	<code>true</code> - Prefer the language keyword for local variables, method parameters, and class members, instead of the type name, for types that have a keyword to represent them <code>false</code> - Prefer the type name for local variables, method parameters, and class members, instead of the language keyword
Visual Studio default	<code>true:silent</code>

Code examples:

```
// dotnet_style_predefined_type_for_locals_parameters_members = true
private int _member;

// dotnet_style_predefined_type_for_locals_parameters_members = false
private Int32 _member;
```

```

' dotnet_style_predefined_type_for_locals_parameters_members = true
Private _member As Integer

' dotnet_style_predefined_type_for_locals_parameters_members = false
Private _member As Int32

```

dotnet_style_predefined_type_for_member_access

Rule name	dotnet_style_predefined_type_for_member_access
Rule ID	IDE0013 and IDE0015
Applicable languages	C# and Visual Basic
Values	<p><code>true</code> - Prefer the language keyword for member access expressions, instead of the type name, for types that have a keyword to represent them</p> <p><code>false</code> - Prefer the type name for member access expressions, instead of the language keyword</p>
Visual Studio default	<code>true:silent</code>

Code examples:

```

// dotnet_style_predefined_type_for_member_access = true
var local = int.MaxValue;

// dotnet_style_predefined_type_for_member_access = false
var local = Int32.MaxValue;

```

```

' dotnet_style_predefined_type_for_member_access = true
Dim local = Integer.MaxValue

' dotnet_style_predefined_type_for_member_access = false
Dim local = Int32.MaxValue

```

Modifier preferences

The style rules in this section concern modifier preferences, including requiring accessibility modifiers, specifying the desired modifier sort order, and requiring the read-only modifier.

These rules could appear in an `.editorconfig` file as follows:

```

# CSharp and Visual Basic code style settings:
[*.cs,vb]
dotnet_style_require_accessibility_modifiers = always:suggestion
dotnet_style_READONLY_field = true:warning

# CSharp code style settings:
[*.cs]
csharp_preferred_modifier_order =
public,private,protected,internal,static,extern,new,virtual,abstract,sealed,override,readonly,unsafe,volatile,
async:suggestion

# Visual Basic code style settings:
[*.vb]
visual_basic_preferred_modifier_order =
Partial,Default,Private,Protected,Public,Friend,NotOverridable,Overridable,MustOverride,Overloads,Overrides,Mu
stInherit,NotInheritable,Static,Shared,Shadows,ReadOnly,WriteOnly,Dim,Const,WithEvents,Widening,Narrowing,Cust
om,Async:suggestion

```

dotnet_style_require_accessibility_modifiers

Rule name	dotnet_style_require_accessibility_modifiers
Rule ID	IDE0040
Applicable languages	C# and Visual Basic
Values	<p><code>always</code> - Prefer accessibility modifiers to be specified.</p> <p><code>for_non_interface_members</code> - Prefer accessibility modifiers to be declared except for public interface members. (This is the same as always and has been added for future-proofing if C# adds default interface methods.)</p> <p><code>never</code> - Do not prefer accessibility modifiers to be specified.</p> <p><code>omit_if_default</code> - Prefer accessibility modifiers to be specified except if they are the default modifier.</p>
Visual Studio default	<code>for_non_interface_members:silent</code>
Introduced version	Visual Studio 2017 version 15.5

Code examples:

```

// dotnet_style_require_accessibility_modifiers = always
// dotnet_style_require_accessibility_modifiers = for_non_interface_members
class MyClass
{
    private const string thisFieldIsConst = "constant";
}

// dotnet_style_require_accessibility_modifiers = never
class MyClass
{
    const string thisFieldIsConst = "constant";
}

```

csharp_preferred_modifier_order

Rule name	csharp_preferred_modifier_order
Rule ID	IDE0036
Applicable languages	C#
Values	One or more C# modifiers, such as <code>public</code> , <code>private</code> , and <code>protected</code>
Visual Studio default	<code>public, private, protected, internal, static, extern, new, virtual, abstract, sealed, override, readonly, unsafe, volatile, async:silent</code>
Introduced version	Visual Studio 2017 version 15.5

- When this rule is set to a list of modifiers, prefer the specified ordering.
- When this rule is omitted from the file, do not prefer a modifier order.

Code examples:

```
// csharp_preferred_modifier_order =
public,private,protected,internal,static,extern,new,virtual,abstract,sealed,override,readonly,unsafe,volatile,
async
class MyClass
{
    private static readonly int _daysInYear = 365;
}
```

visual_basic_preferred_modifier_order

Rule name	visual_basic_preferred_modifier_order
Rule ID	IDE0036
Applicable languages	Visual Basic
Values	One or more Visual Basic modifiers, such as <code>Partial</code> , <code>Private</code> , and <code>Public</code>
Visual Studio default	<code>Partial, Default, Private, Protected, Public, Friend, NotOverridable, Overridable, MustOverride, Overloads, Overrides, MustInherit, NotInheritable, Static, Shared, Shadows, ReadOnly, WriteOnly, Dim, Const, WithEvents, Widening, Narrowing, Custom, Async:silent</code>
Introduced version	Visual Studio 2017 version 15.5

- When this rule is set to a list of modifiers, prefer the specified ordering.
- When this rule is omitted from the file, do not prefer a modifier order.

Code examples:

```

' visual_basic_preferred_modifier_order =
Partial,Default,Private,Protected,Public,Friend,NotOverridable,Overridable,MustOverride,Overloads,Overrides,
stInherit,NotInheritable,Static,Shared,Shadows,ReadOnly,WriteOnly,Dim,Const,WithEvents,Widening,Narrowing,Cust
om,Async
Public Class MyClass
    Private Shared ReadOnly daysInYear As Int = 365
End Class

```

dotnet_style_READONLY_field

Rule name	dotnet_style_READONLY_field
Rule ID	IDE0044
Applicable languages	C# and Visual Basic
Values	<p><code>true</code> - Prefer that fields should be marked with <code>readonly</code> (C#) or <code>ReadOnly</code> (Visual Basic) if they are only ever assigned inline, or inside of a constructor</p> <p><code>false</code> - Specify no preference over whether fields should be marked with <code>readonly</code> (C#) or <code>ReadOnly</code> (Visual Basic)</p>
Visual Studio default	<code>true:suggestion</code>
Introduced version	Visual Studio 2017 version 15.7

Code examples:

```

// dotnet_style_READONLY_field = true
class MyClass
{
    private readonly int _daysInYear = 365;
}

```

```

' dotnet_style_READONLY_field = true
Public Class MyClass
    Private ReadOnly daysInYear As Int = 365
End Class

```

Parentheses preferences

The style rules in this section concern parentheses preferences, including the use of parentheses for arithmetic, relational, and other binary operators.

These rules could appear in an `.editorconfig` file as follows:

```

# CSharp and Visual Basic code style settings:
[*.{cs,vb}]
dotnet_style_parentheses_in_arithmetic_binary_operators = always_for_clarity:silent
dotnet_style_parentheses_in_relational_binary_operators = always_for_clarity:silent
dotnet_style_parentheses_in_other_binary_operators = always_for_clarity:silent
dotnet_style_parentheses_in_other_operators = never_if_unnecessary:silent

```

dotnet_style_parentheses_in_arithmetic_binary_operators

Rule name	dotnet_style_parentheses_in_arithmetic_binary_operators
Rule ID	IDE0047
Applicable languages	C# and Visual Basic
Values	<p><code>always_for_clarity</code> - Prefer parentheses to clarify arithmetic operator (<code>*</code>, <code>/</code>, <code>%</code>, <code>+</code>, <code>-</code>, <code><<</code>, <code>>></code>, <code>&</code>, <code>^</code>, <code> </code>) precedence</p> <p><code>never_if_unnecessary</code> - Prefer to not have parentheses when arithmetic operator (<code>*</code>, <code>/</code>, <code>%</code>, <code>+</code>, <code>-</code>, <code><<</code>, <code>>></code>, <code>&</code>, <code>^</code>, <code> </code>) precedence is obvious</p>
Visual Studio default	<code>always_for_clarity:silent</code>
Introduced version	Visual Studio 2017 version 15.8

Code examples:

```
// dotnet_style_parentheses_in_arithmetic_binary_operators = always_for_clarity
var v = a + (b * c);

// dotnet_style_parentheses_in_arithmetic_binary_operators = never_if_unnecessary
var v = a + b * c;
```

```
' dotnet_style_parentheses_in_arithmetic_binary_operators = always_for_clarity
Dim v = a + (b * c)

' dotnet_style_parentheses_in_arithmetic_binary_operators = never_if_unnecessary
Dim v = a + b * c
```

dotnet_style_parentheses_in_relational_binary_operators

Rule name	dotnet_style_parentheses_in_relational_binary_operators
Rule ID	IDE0047
Applicable languages	C# and Visual Basic
Values	<p><code>always_for_clarity</code> - Prefer parentheses to clarify relational operator (<code>></code>, <code><</code>, <code><=</code>, <code>>=</code>, <code>is</code>, <code>as</code>, <code>==</code>, <code>!=</code>) precedence</p> <p><code>never_if_unnecessary</code> - Prefer to not have parentheses when relational operator (<code>></code>, <code><</code>, <code><=</code>, <code>>=</code>, <code>is</code>, <code>as</code>, <code>==</code>, <code>!=</code>) precedence is obvious</p>
Visual Studio default	<code>always_for_clarity:silent</code>
Introduced version	Visual Studio 2017 version 15.8

Code examples:

```
// dotnet_style_parentheses_in_relational_binary_operators = always_for_clarity
var v = (a < b) == (c > d);

// dotnet_style_parentheses_in_relational_binary_operators = never_if_unnecessary
var v = a < b == c > d;
```

```
' dotnet_style_parentheses_in_relational_binary_operators = always_for_clarity
Dim v = (a < b) = (c > d)

' dotnet_style_parentheses_in_relational_binary_operators = never_if_unnecessary
Dim v = a < b = c > d
```

dotnet_style_parentheses_in_other_binary_operators

Rule name	dotnet_style_parentheses_in_other_binary_operators
Rule ID	IDE0047
Applicable languages	C# and Visual Basic
Values	<code>always_for_clarity</code> - Prefer parentheses to clarify other binary operator (<code>&&</code> , <code> </code> , <code>??</code>) precedence <code>never_if_unnecessary</code> - Prefer to not have parentheses when other binary operator (<code>&&</code> , <code> </code> , <code>??</code>) precedence is obvious
Visual Studio default	<code>always_for_clarity:silent</code>
Introduced version	Visual Studio 2017 version 15.8

Code examples:

```
// dotnet_style_parentheses_in_other_binary_operators = always_for_clarity
var v = a || (b && c);

// dotnet_style_parentheses_in_other_binary_operators = never_if_unnecessary
var v = a || b && c;
```

```
' dotnet_style_parentheses_in_other_binary_operators = always_for_clarity
Dim v = a OrElse (b AndAlso c)

' dotnet_style_parentheses_in_other_binary_operators = never_if_unnecessary
Dim v = a OrElse b AndAlso c
```

dotnet_style_parentheses_in_other_operators

Rule name	dotnet_style_parentheses_in_other_operators
Rule ID	IDE0047

Applicable languages	C# and Visual Basic
Values	<p><code>always_for_clarity</code> - Prefer parentheses to clarify operator precedence</p> <p><code>never_if_unnecessary</code> - Prefer to not have parentheses when operator precedence is obvious</p>
Visual Studio default	<code>never_if_unnecessary:silent</code>
Introduced version	Visual Studio 2017 version 15.8

Code examples:

```
// dotnet_style_parentheses_in_other_operators = always_for_clarity
var v = (a.b).Length;

// dotnet_style_parentheses_in_other_operators = never_if_unnecessary
var v = a.b.Length;
```

```
' dotnet_style_parentheses_in_other_operators = always_for_clarity
Dim v = (a.b).Length

' dotnet_style_parentheses_in_other_operators = never_if_unnecessary
Dim v = a.b.Length
```

Expression-level preferences

The style rules in this section concern expression-level preferences, including the use of object initializers, collection initializers, explicit or inferred tuple names, and inferred anonymous types.

These rules could appear in an `.editorconfig` file as follows:

```
# CSharp and Visual Basic code style settings:
[*.{cs,vb}]
dotnet_style_object_initializer = true:suggestion
dotnet_style_collection_initializer = true:suggestion
dotnet_style_explicit_tuple_names = true:suggestion
dotnet_style_prefer_inferred_tuple_names = true:suggestion
dotnet_style_prefer_inferred_anonymous_type_member_names = true:suggestion
dotnet_style_prefer_auto_properties = true:silent
dotnet_style_prefer_conditional_expression_over_assignment = true:suggestion
dotnet_style_prefer_conditional_expression_over_return = true:suggestion
dotnet_style_prefer_compound_assignment = true:suggestion
```

dotnet_style_object_initializer

Rule name	dotnet_style_object_initializer
Rule ID	IDE0017
Applicable languages	C# and Visual Basic

Values	<input type="checkbox"/> <code>true</code> - Prefer objects to be initialized using object initializers when possible <input type="checkbox"/> <code>false</code> - Prefer objects to <i>not</i> be initialized using object initializers
Visual Studio default	<code>true:suggestion</code>

Code examples:

```
// dotnet_style_object_initializer = true
var c = new Customer() { Age = 21 };

// dotnet_style_object_initializer = false
var c = new Customer();
c.Age = 21;
```

```
' dotnet_style_object_initializer = true
Dim c = New Customer() With {.Age = 21}

' dotnet_style_object_initializer = false
Dim c = New Customer()
c.Age = 21
```

dotnet_style_collection_initializer

Rule name	dotnet_style_collection_initializer
Rule ID	IDE0028
Applicable languages	C# and Visual Basic
Values	<input type="checkbox"/> <code>true</code> - Prefer collections to be initialized using collection initializers when possible <input type="checkbox"/> <code>false</code> - Prefer collections to <i>not</i> be initialized using collection initializers
Visual Studio default	<code>true:suggestion</code>

Code examples:

```
// dotnet_style_collection_initializer = true
var list = new List<int> { 1, 2, 3 };

// dotnet_style_collection_initializer = false
var list = new List<int>();
list.Add(1);
list.Add(2);
list.Add(3);
```

```

' dotnet_style_collection_initializer = true
Dim list = New List(Of Integer) From {1, 2, 3}

' dotnet_style_collection_initializer = false
Dim list = New List(Of Integer)
list.Add(1)
list.Add(2)
list.Add(3)

```

dotnet_style_explicit_tuple_names

Rule name	dotnet_style_explicit_tuple_names
Rule ID	IDE0033
Applicable languages	C# 7.0+ and Visual Basic 15+
Values	<input type="checkbox"/> true - Prefer tuple names to ItemX properties <input type="checkbox"/> false - Prefer ItemX properties to tuple names
Visual Studio default	<input type="checkbox"/> true:suggestion

Code examples:

```

// dotnet_style_explicit_tuple_names = true
(string name, int age) customer = GetCustomer();
var name = customer.name;

// dotnet_style_explicit_tuple_names = false
(string name, int age) customer = GetCustomer();
var name = customer.Item1;

```

```

' dotnet_style_explicit_tuple_names = true
Dim customer As (name As String, age As Integer) = GetCustomer()
Dim name = customer.name

' dotnet_style_explicit_tuple_names = false
Dim customer As (name As String, age As Integer) = GetCustomer()
Dim name = customer.Item1

```

dotnet_style_prefer_inferred_tuple_names

Rule name	dotnet_style_prefer_inferred_tuple_names
Rule ID	IDE0037
Applicable languages	C# 7.1+ and Visual Basic 15+
Values	<input type="checkbox"/> true - Prefer inferred tuple element names <input type="checkbox"/> false - Prefer explicit tuple element names

Visual Studio default	<code>true:suggestion</code>
Introduced version	Visual Studio 2017 version 15.6

Code examples:

```
// dotnet_style_prefer_inferred_tuple_names = true
var tuple = (age, name);

// dotnet_style_prefer_inferred_tuple_names = false
var tuple = (age: age, name: name);
```

```
' dotnet_style_prefer_inferred_tuple_names = true
Dim tuple = (name, age)

' dotnet_style_prefer_inferred_tuple_names = false
Dim tuple = (name:=name, age:=age)
```

dotnet_style_prefer_inferred_anonymous_type_member_names

Rule name	dotnet_style_prefer_inferred_anonymous_type_member_names
Rule ID	IDE0037
Applicable languages	C# and Visual Basic
Values	<input type="checkbox"/> <code>true</code> - Prefer inferred anonymous type member names <input type="checkbox"/> <code>false</code> - Prefer explicit anonymous type member names
Visual Studio default	<code>true:suggestion</code>
Introduced version	Visual Studio 2017 version 15.6

Code examples:

```
// dotnet_style_prefer_inferred_anonymous_type_member_names = true
var anon = new { age, name };

// dotnet_style_prefer_inferred_anonymous_type_member_names = false
var anon = new { age = age, name = name };
```

```
' dotnet_style_prefer_inferred_anonymous_type_member_names = true
Dim anon = New With {name, age}

' dotnet_style_prefer_inferred_anonymous_type_member_names = false
Dim anon = New With {.name = name, .age = age}
```

dotnet_style_prefer_auto_properties

Rule name	dotnet_style_prefer_auto_properties
Rule ID	IDE0032
Applicable languages	C# and Visual Basic
Values	<p><code>true</code> - Prefer autoproperties over properties with private backing fields</p> <p><code>false</code> - Prefer properties with private backing fields over autoproperties</p>
Visual Studio default	<code>true:suggestion</code>
Introduced version	Visual Studio 2017 version 15.7

Code examples:

```
// dotnet_style_prefer_auto_properties = true
private int Age { get; }

// dotnet_style_prefer_auto_properties = false
private int age;

public int Age
{
    get
    {
        return age;
    }
}
```

```
' dotnet_style_prefer_auto_properties = true
Public ReadOnly Property Age As Integer

' dotnet_style_prefer_auto_properties = false
Private _age As Integer

Public ReadOnly Property Age As Integer
    Get
        Return _age
    End Get
End Property
```

dotnet_style_prefer_is_null_check_over_reference_equality_method

Rule name	dotnet_style_prefer_is_null_check_over_reference_equality_method
Rule ID	IDE0041
Applicable languages	C# and Visual Basic

Values	<p><code>true</code> - Prefer using a null check with pattern-matching over <code>object.ReferenceEquals</code></p> <p><code>false</code> - Prefer <code>object.ReferenceEquals</code> over a null check with pattern-matching</p>
Visual Studio default	<code>true:suggestion</code>
Introduced version	Visual Studio 2017 version 15.7

Code examples:

```
// dotnet_style_prefer_is_null_check_over_reference_equality_method = true
if (value is null)
    return;

// dotnet_style_prefer_is_null_check_over_reference_equality_method = false
if (object.ReferenceEquals(value, null))
    return;
```

```
' dotnet_style_prefer_is_null_check_over_reference_equality_method = true
If value Is Nothing
    Return
End If

' dotnet_style_prefer_is_null_check_over_reference_equality_method = false
If Object.ReferenceEquals(value, Nothing)
    Return
End If
```

dotnet_style_prefer_conditional_expression_over_assignment

Rule name	dotnet_style_prefer_conditional_expression_over_assignment
Rule ID	IDE0045
Applicable languages	C# and Visual Basic
Values	<p><code>true</code> - Prefer assignments with a ternary conditional over an if-else statement</p> <p><code>false</code> - Prefer assignments with an if-else statement over a ternary conditional</p>
Visual Studio default	<code>true:suggestion</code>
Introduced version	Visual Studio 2017 version 15.8

Code examples:

```
// dotnet_style_prefer_conditional_expression_over_assignment = true
string s = expr ? "hello" : "world";

// dotnet_style_prefer_conditional_expression_over_assignment = false
string s;
if (expr)
{
    s = "hello";
}
else
{
    s = "world";
}
```

```
' dotnet_style_prefer_conditional_expression_over_assignment = true
Dim s As String = If(expr, "hello", "world")

' dotnet_style_prefer_conditional_expression_over_assignment = false
Dim s As String
If expr Then
    s = "hello"
Else
    s = "world"
End If
```

dotnet_style_prefer_conditional_expression_over_return

Rule name	dotnet_style_prefer_conditional_expression_over_return
Rule ID	IDE0046
Applicable languages	C# and Visual Basic
Values	<p><code>true</code> - Prefer return statements to use a ternary conditional over an if-else statement</p> <p><code>false</code> - Prefer return statements to use an if-else statement over a ternary conditional</p>
Visual Studio default	<code>true:suggestion</code>
Introduced version	Visual Studio 2017 version 15.8

Code examples:

```
// dotnet_style_prefer_conditional_expression_over_return = true
return expr ? "hello" : "world"

// dotnet_style_prefer_conditional_expression_over_return = false
if (expr)
{
    return "hello";
}
else
{
    return "world";
}
```

```

' dotnet_style_prefer_conditional_expression_over_return = true
Return If(expr, "hello", "world")

' dotnet_style_prefer_conditional_expression_over_return = false
If expr Then
    Return "hello"
Else
    Return "world"
End If

```

dotnet_style_prefer_compound_assignment

Rule name	dotnet_style_prefer_compound_assignment
Rule ID	IDE0054
Applicable languages	C# and Visual Basic
Values	<input type="checkbox"/> true - Prefer compound assignment expressions <input type="checkbox"/> false - Don't prefer compound assignment expressions
Visual Studio default	<input type="checkbox"/> true:suggestion

Code examples:

```

// dotnet_style_prefer_compound_assignment = true
x += 1;

// dotnet_style_prefer_compound_assignment = false
x = x + 1;

```

```

' dotnet_style_prefer_compound_assignment = true
x += 1

' dotnet_style_prefer_compound_assignment = false
x = x + 1

```

Null-checking preferences

The style rules in this section concern null-checking preferences.

These rules could appear in an `.editorconfig` file as follows:

```

# CSharp and Visual Basic code style settings:
[*.{cs,vb}]
dotnet_style_coalesce_expression = true:suggestion
dotnet_style_null_propagation = true:suggestion

```

dotnet_style_coalesce_expression

Rule name	dotnet_style_coalesce_expression
Rule ID	IDE0029

Applicable languages	C# and Visual Basic
Values	<p><code>true</code> - Prefer null coalescing expressions to ternary operator checking</p> <p><code>false</code> - Prefer ternary operator checking to null coalescing expressions</p>
Visual Studio default	<code>true:suggestion</code>

Code examples:

```
// dotnet_style_coalesce_expression = true
var v = x ?? y;

// dotnet_style_coalesce_expression = false
var v = x != null ? x : y; // or
var v = x == null ? y : x;
```

```
' dotnet_style_coalesce_expression = true
Dim v = If(x, y)

' dotnet_style_coalesce_expression = false
Dim v = If(x Is Nothing, y, x) ' or
Dim v = If(x IsNot Nothing, x, y)
```

dotnet_style_null_propagation

Rule name	dotnet_style_null_propagation
Rule ID	IDE0031
Applicable languages	C# 6.0+ and Visual Basic 14+
Values	<p><code>true</code> - Prefer to use null-conditional operator when possible</p> <p><code>false</code> - Prefer to use ternary null checking where possible</p>
Visual Studio default	<code>true:suggestion</code>

Code examples:

```
// dotnet_style_null_propagation = true
var v = o?.ToString();

// dotnet_style_null_propagation = false
var v = o == null ? null : o.ToString(); // or
var v = o != null ? o.String() : null;
```

```

' dotnet_style_null_propagation = true
Dim v = o?.ToString()

' dotnet_style_null_propagation = false
Dim v = If(o Is Nothing, Nothing, o.ToString()) ' or
Dim v = If(o IsNot Nothing, o.ToString(), Nothing)

```

.NET code quality settings

The quality rules in this section apply to both C# and Visual Basic code. They're used to configure code analyzers that are built into the Visual Studio integrated development environment (IDE). For information about configuring FxCop analyzers with an EditorConfig file, see [Configure FxCop analyzers](#).

- [Parameter preferences](#)
 - `dotnet_code_quality_unused_parameters`

Parameter preferences

The quality rules in this section concern method parameters.

These rules could appear in an `.editorconfig` file as follows:

```

# CSharp and Visual Basic code quality settings:
[*.{cs,vb}]
dotnet_code_quality_unused_parameters = all:suggestion

```

`dotnet_code_quality_unused_parameters`

Rule name	<code>dotnet_code_quality_unused_parameters</code>
Rule ID	IDE0060
Applicable languages	C# and Visual Basic
Values	<p><code>all</code> - Flag methods with any accessibility that contain unused parameters</p> <p><code>non_public</code> - Flag only non-public methods that contain unused parameters</p>
Visual Studio default	<code>all:suggestion</code>

Code examples:

```

// dotnet_code_quality_unused_parameters = all:suggestion
public int GetNum() { return 1; }

// dotnet_code_quality_unused_parameters = non_public:suggestion
public int GetNum(int arg1) { return 1; }

```

```

' dotnet_code_quality_unused_parameters = all:suggestion
Public Function GetNum()
    Return 1
End Function

' dotnet_code_quality_unused_parameters = non_public:suggestion
Public Function GetNum(arg1 As Integer)
    Return 1
End Function

```

C# code style settings

The style rules in this section are applicable to C# only.

- [Implicit and explicit types](#)
 - csharp_style_var_for_builtin_types
 - csharp_style_var_when_type_is_apparent
 - csharp_style_var_elsewhere
- [Expression-bodied members](#)
 - csharp_style_expression_bodied_methods
 - csharp_style_expression_bodied_constructors
 - csharp_style_expression_bodied_operators
 - csharp_style_expression_bodied_properties
 - csharp_style_expression_bodied_indexers
 - csharp_style_expression_bodied_accessors
 - csharp_style_expression_bodied_lambdas
 - csharp_style_expression_bodied_local_functions
- [Pattern matching](#)
 - csharp_style_pattern_matching_over_is_with_cast_check
 - csharp_style_pattern_matching_over_as_with_null_check
- [Inlined variable declarations](#)
 - csharp_style_inlined_variable_declaration
- [Expression-level preferences](#)
 - csharp_prefer_simple_default_expression
- ["Null" checking preferences](#)
 - csharp_style_throw_expression
 - csharp_style_conditional_delegate_call
- [Code block preferences](#)
 - csharp_prefer_braces
- [Unused value preferences](#)
 - csharp_style_unused_value_expression_statement_preference
 - csharp_style_unused_value_assignment_preference
- [Index and range preferences](#)
 - csharp_style_prefer_index_operator
 - csharp_style_prefer_range_operator
- [Miscellaneous preferences](#)
 - csharp_style_deconstructed_variable_declaration
 - csharp_style_pattern_local_over_anonymous_function
 - csharp_using_directive_placement

- csharp_prefer_static_local_function
- csharp_prefer_simple_using_statement
- csharp_style_prefer_switch_expression

Implicit and explicit types

The style rules in this section concern the use of the `var` keyword versus an explicit type in a variable declaration. This rule can be applied separately to built-in types, when the type is apparent, and elsewhere.

Example `.editorconfig` file:

```
# CSharp code style settings:
[*.cs]
csharp_style_var_for_builtin_types = true:suggestion
csharp_style_var_when_type_is_apparent = true:suggestion
csharp_style_var_elsewhere = true:suggestion
```

csharp_style_var_for_builtin_types

Rule name	csharp_style_var_for_builtin_types
Rule ID	IDE0007 and IDE0008
Applicable languages	C#
Values	<p><code>true</code> - Prefer <code>var</code> is used to declare variables with built-in system types such as <code>int</code></p> <p><code>false</code> - Prefer explicit type over <code>var</code> to declare variables with built-in system types such as <code>int</code></p>
Visual Studio default	<code>true:silent</code>

Code examples:

```
// csharp_style_var_for_builtin_types = true
var x = 5;

// csharp_style_var_for_builtin_types = false
int x = 5;
```

csharp_style_var_when_type_is_apparent

Rule name	csharp_style_var_when_type_is_apparent
Rule ID	IDE0007 and IDE0008
Applicable languages	C#

Values

`true` - Prefer `var` when the type is already mentioned on the right-hand side of a declaration expression

`false` - Prefer explicit type over `var` when the type is already mentioned on the right-hand side of a declaration expression

Visual Studio default

`true:silent`

Code examples:

```
// csharp_style_var_when_type_is_apparent = true  
var obj = new Customer();  
  
// csharp_style_var_when_type_is_apparent = false  
Customer obj = new Customer();
```

csharp_style_var_elsewhere**Rule name**

csharp_style_var_elsewhere

Rule ID

IDE0007 and IDE0008

Applicable languages

C#

Values

`true` - Prefer `var` over explicit type in all cases, unless overridden by another code style rule

`false` - Prefer explicit type over `var` in all cases, unless overridden by another code style rule

Visual Studio default

`true:silent`

Code examples:

```
// csharp_style_var_elsewhere = true  
var f = this.Init();  
  
// csharp_style_var_elsewhere = false  
bool f = this.Init();
```

Expression-bodied members

The style rules in this section concern the use of [expression-bodied members](#) when the logic consists of a single expression. This rule can be applied to methods, constructors, operators, properties, indexers, and accessors.

Example `.editorconfig` file:

```

# CSharp code style settings:
[*.cs]
csharp_style_expression_bodied_methods = false:silent
csharp_style_expression_bodied_constructors = false:silent
csharp_style_expression_bodied_operators = false:silent
csharp_style_expression_bodied_properties = true:suggestion
csharp_style_expression_bodied_indexers = true:suggestion
csharp_style_expression_bodied_accessors = true:suggestion
csharp_style_expression_bodied_lambdas = true:silent
csharp_style_expression_bodied_local_functions = false:silent

```

csharp_style_expression_bodied_methods

Rule name	csharp_style_expression_bodied_methods
Rule ID	IDE0022
Applicable languages	C# 6.0+
Values	<div style="display: flex; justify-content: space-between;"> <div style="flex: 1;"> true - Prefer expression bodies for methods </div> <div style="flex: 1;"> when_on_single_line - Prefer expression bodies for methods when they will be a single line </div> <div style="flex: 1;"> false - Prefer block bodies for methods </div> </div>
Visual Studio default	false:silent

Code examples:

```

// csharp_style_expression_bodied_methods = true
public int GetAge() => this.Age;

// csharp_style_expression_bodied_methods = false
public int GetAge() { return this.Age; }

```

csharp_style_expression_bodied_constructors

Rule name	csharp_style_expression_bodied_constructors
Rule ID	IDE0021
Applicable languages	C# 7.0+
Values	<div style="display: flex; justify-content: space-between;"> <div style="flex: 1;"> true - Prefer expression bodies for constructors </div> <div style="flex: 1;"> when_on_single_line - Prefer expression bodies for constructors when they will be a single line </div> <div style="flex: 1;"> false - Prefer block bodies for constructors </div> </div>
Visual Studio default	false:silent

Code examples:

```
// csharp_style_expression_bodied_constructors = true
public Customer(int age) => Age = age;

// csharp_style_expression_bodied_constructors = false
public Customer(int age) { Age = age; }
```

csharp_style_expression_bodied_operators

Rule name	csharp_style_expression_bodied_operators
Rule ID	IDE0023 and IDE0024
Applicable languages	C# 7.0+
Values	<p><code>true</code> - Prefer expression bodies for operators</p> <p><code>when_on_single_line</code> - Prefer expression bodies for operators when they will be a single line</p> <p><code>false</code> - Prefer block bodies for operators</p>
Visual Studio default	<code>false:silent</code>

Code examples:

```
// csharp_style_expression_bodied_operators = true
public static ComplexNumber operator + (ComplexNumber c1, ComplexNumber c2)
    => new ComplexNumber(c1.Real + c2.Real, c1.Imaginary + c2.Imaginary);

// csharp_style_expression_bodied_operators = false
public static ComplexNumber operator + (ComplexNumber c1, ComplexNumber c2)
{ return new ComplexNumber(c1.Real + c2.Real, c1.Imaginary + c2.Imaginary); }
```

csharp_style_expression_bodied_properties

Rule name	csharp_style_expression_bodied_properties
Rule ID	IDE0025
Applicable languages	C# 7.0+
Values	<p><code>true</code> - Prefer expression bodies for properties</p> <p><code>when_on_single_line</code> - Prefer expression bodies for properties when they will be a single line</p> <p><code>false</code> - Prefer block bodies for properties</p>
Visual Studio default	<code>true:silent</code>

Code examples:

```
// csharp_style_expression_bodied_properties = true
public int Age => _age;

// csharp_style_expression_bodied_properties = false
public int Age { get { return _age; } }
```

csharp_style_expression_bodied_indexers

Rule name	csharp_style_expression_bodied_indexers
Rule ID	IDE0026
Applicable languages	C# 7.0+
Values	<div style="display: flex; justify-content: space-between;"> <div style="flex: 1;"><code>true</code></div> <div>- Prefer expression bodies for indexers</div> </div> <div style="margin-top: 10px;"> <div style="display: flex; justify-content: space-between;"> <div style="flex: 1;"><code>when_on_single_line</code></div> <div>- Prefer expression bodies for indexers when they will be a single line</div> </div> <div style="margin-top: 10px;"> <div style="display: flex; justify-content: space-between;"> <div style="flex: 1;"><code>false</code></div> <div>- Prefer block bodies for indexers</div> </div> </div> </div>
Visual Studio default	<code>true:silent</code>

Code examples:

```
// csharp_style_expression_bodied_indexers = true
public T this[int i] => _values[i];

// csharp_style_expression_bodied_indexers = false
public T this[int i] { get { return _values[i]; } }
```

csharp_style_expression_bodied_accessors

Rule name	csharp_style_expression_bodied_accessors
Rule ID	IDE0027
Applicable languages	C# 7.0+
Values	<div style="display: flex; justify-content: space-between;"> <div style="flex: 1;"><code>true</code></div> <div>- Prefer expression bodies for accessors</div> </div> <div style="margin-top: 10px;"> <div style="display: flex; justify-content: space-between;"> <div style="flex: 1;"><code>when_on_single_line</code></div> <div>- Prefer expression bodies for accessors when they will be a single line</div> </div> <div style="margin-top: 10px;"> <div style="display: flex; justify-content: space-between;"> <div style="flex: 1;"><code>false</code></div> <div>- Prefer block bodies for accessors</div> </div> </div> </div>
Visual Studio default	<code>true:silent</code>

Code examples:

```
// csharp_style_expression_bodied_accessors = true
public int Age { get => _age; set => _age = value; }

// csharp_style_expression_bodied_accessors = false
public int Age { get { return _age; } set { _age = value; } }
```

csharp_style_expression_bodied_lambdas

Rule name	csharp_style_expression_bodied_lambdas
Rule ID	IDE0053
Values	<div style="display: flex; justify-content: space-between;"> <div style="flex: 1;"> <input type="checkbox"/> <code>true</code> - Prefer expression bodies for lambdas </div> <div style="flex: 1;"> <input type="checkbox"/> <code>when_on_single_line</code> - Prefer expression bodies for lambdas when they will be a single line </div> <div style="flex: 1;"> <input type="checkbox"/> <code>false</code> - Prefer block bodies for lambdas </div> </div>
Visual Studio default	<code>true:silent</code>

Code examples:

```
// csharp_style_expression_bodied_lambdas = true
Func<int, int> square = x => x * x;

// csharp_style_expression_bodied_lambdas = false
Func<int, int> square = x => { return x * x; };
```

csharp_style_expression_bodied_local_functions

Starting with C# 7.0, C# supports [local functions](#). Local functions are private methods of a type that are nested in another member.

Rule name	csharp_style_expression_bodied_local_functions
Rule ID	IDE0061
Applicable languages	C# 7.0+
Values	<div style="display: flex; justify-content: space-between;"> <div style="flex: 1;"> <input type="checkbox"/> <code>true</code> - Prefer expression bodies for local functions </div> <div style="flex: 1;"> <input type="checkbox"/> <code>when_on_single_line</code> - Prefer expression bodies for local functions when they will be a single line </div> <div style="flex: 1;"> <input type="checkbox"/> <code>false</code> - Prefer block bodies for local functions </div> </div>
Visual Studio default	<code>false:silent</code>

Code examples:

```
// csharp_style_expression_bodied_local_functions = true
void M()
{
    Hello();
    void Hello() => Console.WriteLine("Hello");
}

// csharp_style_expression_bodied_local_functions = false
void M()
{
    Hello();
    void Hello()
    {
        Console.WriteLine("Hello");
    }
}
```

Pattern matching

The style rules in this section concern the use of [pattern matching](#) in C#.

Example `.editorconfig` file:

```
# CSharp code style settings:
[*.cs]
csharp_style_pattern_matching_over_is_with_cast_check = true:suggestion
csharp_style_pattern_matching_over_as_with_null_check = true:suggestion
```

csharp_style_pattern_matching_over_is_with_cast_check

Rule name	csharp_style_pattern_matching_over_is_with_cast_check
Rule ID	IDE0020
Applicable languages	C# 7.0+
Values	<p><code>true</code> - Prefer pattern matching instead of <code>is</code> expressions with type casts</p> <p><code>false</code> - Prefer <code>is</code> expressions with type casts instead of pattern matching</p>
Visual Studio default	<code>true:suggestion</code>

Code examples:

```
// csharp_style_pattern_matching_over_is_with_cast_check = true
if (o is int i) {...}

// csharp_style_pattern_matching_over_is_with_cast_check = false
if (o is int) {var i = (int)o; ... }
```

csharp_style_pattern_matching_over_as_with_null_check

Rule name	csharp_style_pattern_matching_over_as_with_null_check
------------------	---

Rule ID	IDE0019
Applicable languages	C# 7.0+
Values	<p><code>true</code> - Prefer pattern matching instead of <code>as</code> expressions with null checks to determine if something is of a particular type</p> <p><code>false</code> - Prefer <code>as</code> expressions with null checks instead of pattern matching to determine if something is of a particular type</p>
Visual Studio default	<code>true:suggestion</code>

Code examples:

```
// csharp_style_pattern_matching_over_as_with_null_check = true
if (o is string s) {...}

// csharp_style_pattern_matching_over_as_with_null_check = false
var s = o as string;
if (s != null) {...}
```

Inlined variable declarations

This style rule concerns whether `out` variables are declared inline or not. Starting in C# 7, you can [declare an out variable in the argument list of a method call](#), rather than in a separate variable declaration.

csharp_style_inlined_variable_declaration

Rule name	csharp_style_inlined_variable_declaration
Rule ID	IDE0018
Applicable languages	C# 7.0+
Values	<p><code>true</code> - Prefer <code>out</code> variables to be declared inline in the argument list of a method call when possible</p> <p><code>false</code> - Prefer <code>out</code> variables to be declared before the method call</p>
Visual Studio default	<code>true:suggestion</code>

Code examples:

```
// csharp_style_inlined_variable_declaration = true
if (int.TryParse(value, out int i) {...}

// csharp_style_inlined_variable_declaration = false
int i;
if (int.TryParse(value, out i) {...}
```

Example `.editorconfig` file:

```
# CSharp code style settings:  
[*.cs]  
csharp_style_inlined_variable_declaration = true:suggestion
```

C# expression-level preferences

The style rules in this section concern expression-level preferences.

Example *.editorconfig* file:

```
# CSharp code style settings:  
[*.cs]  
csharp_prefer_simple_default_expression = true:suggestion
```

csharp_prefer_simple_default_expression

This style rule concerns using the `default` literal for default value expressions when the compiler can infer the type of the expression.

Rule name	csharp_prefer_simple_default_expression
Rule ID	IDE0034
Applicable languages	C# 7.1+
Values	<code>true</code> - Prefer <code>default</code> over <code>default(T)</code> <code>false</code> - Prefer <code>default(T)</code> over <code>default</code>
Visual Studio default	<code>true:suggestion</code>

Code examples:

```
// csharp_prefer_simple_default_expression = true  
void DoWork(CancellationToken cancellationToken = default) { ... }  
  
// csharp_prefer_simple_default_expression = false  
void DoWork(CancellationToken cancellationToken = default(CancellationToken)) { ... }
```

C# null-checking preferences

These style rules concern the syntax around `null` checking, including using `throw` expressions or `throw` statements, and whether to perform a null check or use the conditional coalescing operator (`?.`) when invoking a [lambda expression](#).

Example *.editorconfig* file:

```
# CSharp code style settings:  
[*.cs]  
csharp_style_throw_expression = true:suggestion  
csharp_style_conditional_delegate_call = false:suggestion
```

csharp_style_throw_expression

Rule name	csharp_style_throw_expression
Rule ID	IDE0016
Applicable languages	C# 7.0+
Values	<p><code>true</code> - Prefer to use <code>throw</code> expressions instead of <code>throw</code> statements</p> <p><code>false</code> - Prefer to use <code>throw</code> statements instead of <code>throw</code> expressions</p>
Visual Studio default	<code>true:suggestion</code>

Code examples:

```
// csharp_style_throw_expression = true
this.s = s ?? throw new ArgumentNullException(nameof(s));

// csharp_style_throw_expression = false
if (s == null) { throw new ArgumentNullException(nameof(s)); }
this.s = s;
```

csharp_style_conditional_delegate_call

Rule name	csharp_style_conditional_delegate_call
Rule ID	IDE0041
Applicable languages	C# 6.0+
Values	<p><code>true</code> - refer to use the conditional coalescing operator (<code>?.</code>) when invoking a lambda expression, instead of performing a null check</p> <p><code>false</code> - Prefer to perform a null check before invoking a lambda expression, instead of using the conditional coalescing operator (<code>?.</code>)</p>
Visual Studio default	<code>true:suggestion</code>

Code examples:

```
// csharp_style_conditional_delegate_call = true
func?.Invoke(args);

// csharp_style_conditional_delegate_call = false
if (func != null) { func(args); }
```

Code block preferences

This style rule concerns the use of curly braces `{ }` to surround code blocks.

Example `.editorconfig` file:

```
# CSharp code style settings:
[*.cs]
csharp_prefer_braces = true:silent
```

csharp_prefer_braces

Rule name	csharp_prefer_braces
Rule ID	IDE0011
Applicable languages	C#
Values	<input type="checkbox"/> <code>true</code> - Prefer curly braces even for one line of code <input type="checkbox"/> <code>false</code> - Prefer no curly braces if allowed
Visual Studio default	<input type="checkbox"/> <code>true:silent</code>

Code examples:

```
// csharp_prefer_braces = true
if (test) { this.Display(); }

// csharp_prefer_braces = false
if (test) this.Display();
```

Unused value preferences

These style rules concern unused expressions and value assignments.

Example `.editorconfig` file:

```
# CSharp code style settings:
[*.cs]
csharp_style_unused_value_expression_statement_preference = discard_variable:silent
csharp_style_unused_value_assignment_preference = discard_variable:suggestion
```

csharp_style_unused_value_expression_statement_preference

Rule name	csharp_style_unused_value_expression_statement_preference
Rule ID	IDE0058
Applicable languages	C#
Values	<input type="checkbox"/> <code>discard_variable</code> - Prefer to assign an unused expression to a discard <input type="checkbox"/> <code>unused_local_variable</code> - Prefer to assign an unused expression to a local variable
Visual Studio default	<input type="checkbox"/> <code>discard_variable:silent</code>

Code examples:

```
// Original code:  
System.Convert.ToInt32("35");  
  
// After code fix for IDE0058:  
  
// csharp_style_unused_value_expression_statement_preference = discard_variable  
_ = System.Convert.ToInt32("35");  
  
// csharp_style_unused_value_expression_statement_preference = unused_local_variable  
var unused = Convert.ToInt32("35");
```

csharp_style_unused_value_assignment_preference

Rule name	csharp_style_unused_value_assignment_preference
Rule ID	IDE0059
Applicable languages	C#
Values	<code>discard_variable</code> - Prefer to use a discard when assigning a value that's not used <code>unused_local_variable</code> - Prefer to use a local variable when assigning a value that's not used
Visual Studio default	<code>discard_variable:suggestion</code>

Code examples:

```
// csharp_style_unused_value_assignment_preference = discard_variable  
int GetCount(Dictionary<string, int> wordCount, string searchWord)  
{  
    _ = wordCount.TryGetValue(searchWord, out var count);  
    return count;  
}  
  
// csharp_style_unused_value_assignment_preference = unused_local_variable  
int GetCount(Dictionary<string, int> wordCount, string searchWord)  
{  
    var unused = wordCount.TryGetValue(searchWord, out var count);  
    return count;  
}
```

Index and range preferences

These style rules concern the use of index and range operators, which are available in C# 8.0 and later.

Example `.editorconfig` file:

```
# CSharp code style settings:  
[*.cs]  
csharp_style_prefer_index_operator = true:suggestion  
csharp_style_prefer_range_operator = true:suggestion
```

csharp_style_prefer_index_operator

Rule name	csharp_style_prefer_index_operator
Rule ID	IDE0056
Applicable languages	C# 8.0+
Values	<p><code>true</code> - Prefer to use the <code>^</code> operator when calculating an index from the end of a collection</p> <p><code>false</code> - Don't prefer to use the <code>^</code> operator when calculating an index from the end of a collection</p>
Visual Studio default	<code>true:suggestion</code>

Code examples:

```
// csharp_style_prefer_index_operator = true
string[] names = { "Archimedes", "Pythagoras", "Euclid" };
var index = names[^1];

// csharp_style_prefer_index_operator = false
string[] names = { "Archimedes", "Pythagoras", "Euclid" };
var index = names[names.Length - 1];
```

csharp_style_prefer_range_operator

Rule name	csharp_style_prefer_range_operator
Rule ID	IDE0057
Applicable languages	C# 8.0+
Values	<p><code>true</code> - Prefer to use the range operator <code>..</code> when extracting a "slice" of a collection</p> <p><code>false</code> - Don't prefer to use the range operator <code>..</code> when extracting a "slice" of a collection</p>
Visual Studio default	<code>true:suggestion</code>

Code examples:

```
// csharp_style_prefer_range_operator = true
string sentence = "the quick brown fox";
var sub = sentence[0..^4];

// csharp_style_prefer_range_operator = false
string sentence = "the quick brown fox";
var sub = sentence.Substring(0, sentence.Length - 4);
```

Miscellaneous preferences

This section contains miscellaneous style rules.

Example `.editorconfig` file:

```
# CSharp code style settings:  
[*.cs]  
csharp_style_deconstructed_variable_declaration = true:suggestion  
csharp_style_pattern_local_over_anonymous_function = true:suggestion  
csharp_using_directive_placement = outside_namespace:silent  
csharp_prefer_static_local_function = true:suggestion  
csharp_prefer_simple_using_statement = true:suggestion  
csharp_style_prefer_switch_expression = true:suggestion
```

csharp_style_deconstructed_variable_declaration

Rule name	csharp_style_deconstructed_variable_declaration
Rule ID	IDE0042
Applicable languages	C# 7.0+
Values	<code>true</code> - Prefer deconstructed variable declaration <code>false</code> - Do not prefer deconstruction in variable declarations
Visual Studio default	<code>true:suggestion</code>

Code examples:

```
// csharp_style_deconstructed_variable_declaration = true  
var (name, age) = GetPersonTuple();  
Console.WriteLine($"{name} {age}");  
  
(int x, int y) = GetPointTuple();  
Console.WriteLine($"{x} {y}");  
  
// csharp_style_deconstructed_variable_declaration = false  
var person = GetPersonTuple();  
Console.WriteLine($"{person.name} {person.age}");  
  
(int x, int y) point = GetPointTuple();  
Console.WriteLine($"{point.x} {point.y}");
```

csharp_style_pattern_local_over_anonymous_function

Starting with C# 7.0, C# supports [local functions](#). Local functions are private methods of a type that are nested in another member.

Rule name	csharp_style_pattern_local_over_anonymous_function
Rule ID	IDE0039
Applicable languages	C# 7.0+

Values	<code>true</code> - Prefer local functions over anonymous functions <code>false</code> - Prefer anonymous functions over local functions
Visual Studio default	<code>true:suggestion</code>

Code examples:

```
// csharp_style_pattern_local_over_anonymous_function = true
int fibonacci(int n)
{
    return n <= 1 ? 1 : fibonacci(n-1) + fibonacci(n-2);
}

// csharp_style_pattern_local_over_anonymous_function = false
Func<int, int> fibonacci = null;
fibonacci = (int n) =>
{
    return n <= 1 ? 1 : fibonacci(n - 1) + fibonacci(n - 2);
};
```

csharp_using_directive_placement

Rule name	csharp_using_directive_placement
Rule ID	IDE0065
Applicable languages	C#
Values	<code>outside_namespace</code> - Prefer <code>using</code> directives to be placed outside the namespace <code>inside_namespace</code> - Prefer <code>using</code> directives to be placed inside the namespace
Visual Studio default	<code>outside_namespace:silent</code>

Code examples:

```
// csharp_using_directive_placement = outside_namespace
using System;

namespace Conventions
{
    ...
}

// csharp_using_directive_placement = inside_namespace
namespace Conventions
{
    using System;
    ...
}
```

csharp_prefer_static_local_function

Rule name	csharp_prefer_static_local_function
Rule ID	IDE0062
Applicable languages	C# 8.0+
Values	<input type="checkbox"/> <code>true</code> - Prefer local functions to be marked <code>static</code> <input type="checkbox"/> <code>false</code> - Don't prefer local functions to be marked <code>static</code>
Visual Studio default	<code>true:suggestion</code>

Code examples:

```
// csharp_prefer_static_local_function = true
void M()
{
    Hello();
    static void Hello()
    {
        Console.WriteLine("Hello");
    }
}

// csharp_prefer_static_local_function = false
void M()
{
    Hello();
    void Hello()
    {
        Console.WriteLine("Hello");
    }
}
```

csharp_prefer_simple_using_statement

Rule name	csharp_prefer_simple_using_statement
Rule ID	IDE0063
Applicable languages	C# 8.0+
Values	<input type="checkbox"/> <code>true</code> - Prefer to use a <i>simple</i> <code>using</code> statement <input type="checkbox"/> <code>false</code> - Don't prefer to use a <i>simple</i> <code>using</code> statement
Visual Studio default	<code>true:suggestion</code>

Code examples:

```
// csharp_prefer_simple_using_statement = true
using var a = b;

// csharp_prefer_simple_using_statement = false
using (var a = b) { }
```

csharp_style_prefer_switch_expression

Rule name	csharp_style_prefer_switch_expression
Rule ID	IDE0066
Applicable languages	C# 8.0+
Values	<p><code>true</code> - Prefer to use a <code>switch</code> expression (introduced with C# 8.0)</p> <p><code>false</code> - Prefer to use a <code>switch statement</code></p>
Visual Studio default	<code>true:suggestion</code>
Introduced version	Visual Studio 2019 version 16.2

Code examples:

```
// csharp_style_prefer_switch_expression = true
return x switch
{
    1 => 1 * 1,
    2 => 2 * 2,
    _ => 0,
};

// csharp_style_prefer_switch_expression = false
switch (x)
{
    case 1:
        return 1 * 1;
    case 2:
        return 2 * 2;
    default:
        return 0;
}
```

See also

- [Formatting conventions](#)
- [Naming conventions](#)
- [.NET coding convention settings for EditorConfig](#)

Formatting conventions

10/18/2019 • 15 minutes to read • [Edit Online](#)

Formatting conventions for EditorConfig for Visual Studio fall into these categories:

- [.NET formatting settings](#)
- [C# formatting settings](#)

Rule format

Rules for formatting conventions have the following format:

```
rule_name = value
```

For many rules, you specify either `true` (prefer this style) or `false` (do not prefer this style) for `value`. For other rules, you specify a value such as `flush_left` or `before_and_after` to describe when and where to apply the rule. You don't specify a severity.

.NET formatting settings

The formatting rules in this section apply to C# and Visual Basic code.

- [Organize usings](#)
 - `dotnet_sort_system_directives_first`
 - `dotnet_separate_import_directive_groups`

Organize using directives

These formatting rules concern the sorting and display of `using` directives and `Imports` statements.

Example `.editorconfig` file:

```
# .NET formatting settings
[*.{cs,vb}]
dotnet_sort_system_directives_first = true
dotnet_separate_import_directive_groups = true
```

`dotnet_sort_system_directives_first`

Rule name	<code>dotnet_sort_system_directives_first</code>
Applicable languages	C# and Visual Basic
Introduced version	Visual Studio 2017 version 15.3
Values	<p><code>true</code> - Sort System.* <code>using</code> directives alphabetically, and place them before other <code>using</code> directives.</p> <p><code>false</code> - Do not place System.* <code>using</code> directives before other <code>using</code> directives.</p>

Visual Studio default	<code>true</code>
------------------------------	-------------------

Code examples:

```
// dotnet_sort_system_directives_first = true
using System.Collections.Generic;
using System.Threading.Tasks;
using Octokit;

// dotnet_sort_system_directives_first = false
using System.Collections.Generic;
using Octokit;
using System.Threading.Tasks;
```

dotnet_separate_import_directive_groups

Rule name	dotnet_separate_import_directive_groups
Applicable languages	C# and Visual Basic
Introduced version	Visual Studio 2017 version 15.5
Values	<p><code>true</code> - Place a blank line between <code>using</code> directive groups.</p> <p><code>false</code> - Do not place a blank line between <code>using</code> directive groups.</p>
Visual Studio default	<code>false</code>

Code examples:

```
// dotnet_separate_import_directive_groups = true
using System.Collections.Generic;
using System.Threading.Tasks;

using Octokit;

// dotnet_separate_import_directive_groups = false
using System.Collections.Generic;
using System.Threading.Tasks;
using Octokit;
```

C# formatting settings

The formatting rules in this section apply only to C# code.

- [Newline options](#)
 - `csharp_new_line_before_open_brace`
 - `csharp_new_line_before_else`
 - `csharp_new_line_before_catch`
 - `csharp_new_line_before_finally`
 - `csharp_new_line_before_members_in_object_initializers`
 - `csharp_new_line_before_members_in_anonymous_types`

- csharp_new_line_between_query_expression_clauses
- [Indentation options](#)
 - csharp_indent_case_contents
 - csharp_indent_switch_labels
 - csharp_indent_labels
 - csharp_indent_block_contents
 - csharp_indent_braces
 - csharp_indent_case_contents_when_block
- [Spacing options](#)
 - csharp_space_after_cast
 - csharp_space_after_keywords_in_control_flow_statements
 - csharp_space_between_parentheses
 - csharp_space_before_colon_in_inheritance_clause
 - csharp_space_after_colon_in_inheritance_clause
 - csharp_space_around_binary_operators
 - csharp_space_between_method_declaration_parameter_list_parentheses
 - csharp_space_between_method_declaration_empty_parameter_list_parentheses
 - csharp_space_between_method_declaration_name_and_open_parenthesis
 - csharp_space_between_method_call_parameter_list_parentheses
 - csharp_space_between_method_call_empty_parameter_list_parentheses
 - csharp_space_between_method_call_name_and_opening_parenthesis
 - csharp_space_after_comma
 - csharp_space_before_comma
 - csharp_space_after_dot
 - csharp_space_before_dot
 - csharp_space_after_semicolon_in_for_statement
 - csharp_space_before_semicolon_in_for_statement
 - csharp_space_around_declaration_statements
 - csharp_space_before_open_square_brackets
 - csharp_space_between_empty_square_brackets
 - csharp_space_between_square_brackets
- [Wrap options](#)
 - csharp_preserve_single_line_statements
 - csharp_preserve_single_line_blocks

New-line options

These formatting rules concern the use of new lines to format code.

Example `.editorconfig` file:

```
# CSharp formatting settings:
[*.cs]
csharp_new_line_before_open_brace = methods, properties, control_blocks, types
csharp_new_line_before_else = true
csharp_new_line_before_catch = true
csharp_new_line_before_finally = true
csharp_new_line_before_members_in_object_initializers = true
csharp_new_line_before_members_in_anonymous_types = true
csharp_new_line_between_query_expression_clauses = true
```

csharp_new_line_before_open_brace

This rule concerns whether an open brace `{` should be placed on the same line as the preceding code, or on a new line. For this rule, you specify **all**, **none**, or one or more code elements such as **methods** or **properties**, to define when this rule should be applied. To specify multiple code elements, separate them with a comma (,).

Rule name	csharp_new_line_before_open_brace
Applicable languages	C#
Introduced version	Visual Studio 2017 version 15.3
Values	<code>all</code> - Require braces to be on a new line for all expressions ("Allman" style). <code>none</code> - Require braces to be on the same line for all expressions ("K&R"). <code>accessors</code> , <code>anonymous_methods</code> , <code>anonymous_types</code> , <code>control_blocks</code> , <code>events</code> , <code>indexers</code> , <code>lambdas</code> , <code>local_functions</code> , <code>methods</code> , <code>object_collection_array_initializer</code> , <code>properties</code> , <code>types</code> - Require braces to be on a new line for the specified code element ("Allman" style).
Visual Studio default	<code>all</code>

Code examples:

```
// csharp_new_line_before_open_brace = all
void MyMethod()
{
    if (...) {
        ...
    }
}

// csharp_new_line_before_open_brace = none
void MyMethod() {
    if (...) {
        ...
    }
}
```

csharp_new_line_before_else

Rule name	csharp_new_line_before_else
Applicable languages	C#
Introduced version	Visual Studio 2017 version 15.3
Values	<code>true</code> - Place <code>else</code> statements on a new line. <code>false</code> - Place <code>else</code> statements on the same line.

Visual Studio default

true

Code examples:

```
// csharp_new_line_before_else = true
if (...) {
    ...
}
else {
    ...
}

// csharp_new_line_before_else = false
if (...) {
    ...
} else {
    ...
}
```

csharp_new_line_before_catch

Rule name	csharp_new_line_before_catch
Applicable languages	C#
Introduced version	Visual Studio 2017 version 15.3
Values	<div style="display: flex; justify-content: space-between;"><div style="flex: 1;"><p><code>true</code> - Place <code>catch</code> statements on a new line.</p><p><code>false</code> - Place <code>catch</code> statements on the same line.</p></div><div style="flex: 1;"></div></div>
Visual Studio default	true

Code examples:

```
// csharp_new_line_before_catch = true
try {
    ...
}
catch (Exception e) {
    ...
}

// csharp_new_line_before_catch = false
try {
    ...
} catch (Exception e) {
    ...
}
```

csharp_new_line_before_finally

Rule name	csharp_new_line_before_finally
------------------	--------------------------------

Applicable languages	C#
Introduced version	Visual Studio 2017 version 15.3
Values	<p><code>true</code> - Require <code>finally</code> statements to be on a new line after the closing brace.</p> <p><code>false</code> - Require <code>finally</code> statements to be on the same line as the closing brace.</p>
Visual Studio default	<code>true</code>

Code examples:

```
// csharp_new_line_before_finally = true
try {
    ...
}
catch (Exception e) {
    ...
}
finally {
    ...
}

// csharp_new_line_before_finally = false
try {
    ...
} catch (Exception e) {
    ...
} finally {
    ...
}
```

csharp_new_line_before_members_in_object_initializers

Rule name	csharp_new_line_before_members_in_object_initializers
Applicable languages	C#
Introduced version	Visual Studio 2017 version 15.3
Values	<p><code>true</code> - Require members of object initializers to be on separate lines</p> <p><code>false</code> - Require members of object initializers to be on the same line</p>
Visual Studio default	<code>true</code>

Code examples:

```
// csharp_new_line_before_members_in_object_initializers = true
var z = new B()
{
    A = 3,
    B = 4
}

// csharp_new_line_before_members_in_object_initializers = false
var z = new B()
{
    A = 3, B = 4
}
```

csharp_new_line_before_members_in_anonymous_types

Rule name	csharp_new_line_before_members_in_anonymous_types
Applicable languages	C#
Introduced version	Visual Studio 2017 version 15.3
Values	<input type="checkbox"/> true - Require members of anonymous types to be on separate lines <input type="checkbox"/> false - Require members of anonymous types to be on the same line
Visual Studio default	<input checked="" type="checkbox"/> true

Code examples:

```
// csharp_new_line_before_members_in_anonymous_types = true
var z = new
{
    A = 3,
    B = 4
}

// csharp_new_line_before_members_in_anonymous_types = false
var z = new
{
    A = 3, B = 4
}
```

csharp_new_line_between_query_expression_clauses

Rule name	csharp_new_line_between_query_expression_clauses
Applicable languages	C#
Introduced version	Visual Studio 2017 version 15.3

Values	<code>true</code> - Require elements of query expression clauses to be on separate lines <code>false</code> - Require elements of query expression clauses to be on the same line
Visual Studio default	<code>true</code>

Code examples:

```
// csharp_new_line_between_query_expression_clauses = true
var q = from a in e
        from b in e
        select a * b;

// csharp_new_line_between_query_expression_clauses = false
var q = from a in e from b in e
        select a * b;
```

Indentation options

These formatting rules concern the use of indentation to format code.

Example `.editorconfig` file:

```
# CSharp formatting settings:
[*.cs]
csharp_indent_case_contents = true
csharp_indent_switch_labels = true
csharp_indent_labels = flush_left
csharp_indent_block_contents = true
csharp_indent_braces = false
csharp_indent_case_contents_when_block = true
```

csharp_indent_case_contents

Rule name	csharp_indent_case_contents
Applicable languages	C#
Introduced version	Visual Studio 2017 version 15.3
Values	<code>true</code> - Indent <code>switch</code> case contents <code>false</code> - Do not indent <code>switch</code> case contents
Visual Studio default	<code>true</code>

- When this rule is set to **true**, i.
- When this rule is set to **false**, d.

Code examples:

```

// csharp_indent_case_contents = true
switch(c) {
    case Color.Red:
        Console.WriteLine("The color is red");
        break;
    case Color.Blue:
        Console.WriteLine("The color is blue");
        break;
    default:
        Console.WriteLine("The color is unknown.");
        break;
}

// csharp_indent_case_contents = false
switch(c) {
    case Color.Red:
        Console.WriteLine("The color is red");
        break;
    case Color.Blue:
        Console.WriteLine("The color is blue");
        break;
    default:
        Console.WriteLine("The color is unknown.");
        break;
}

```

csharp_indent_switch_labels

Rule name	csharp_indent_switch_labels
Applicable languages	C#
Introduced version	Visual Studio 2017 version 15.3
Values	<input type="checkbox"/> <code>true</code> - Indent <code>switch</code> labels <input type="checkbox"/> <code>false</code> - Do not indent <code>switch</code> labels
Visual Studio default	<input checked="" type="checkbox"/> <code>true</code>

Code examples:

```
// csharp_indent_switch_labels = true
switch(c) {
    case Color.Red:
        Console.WriteLine("The color is red");
        break;
    case Color.Blue:
        Console.WriteLine("The color is blue");
        break;
    default:
        Console.WriteLine("The color is unknown.");
        break;
}

// csharp_indent_switch_labels = false
switch(c) {
case Color.Red:
    Console.WriteLine("The color is red");
    break;
case Color.Blue:
    Console.WriteLine("The color is blue");
    break;
default:
    Console.WriteLine("The color is unknown.");
    break;
}
```

csharp_indent_labels

Rule name	csharp_indent_labels
Applicable languages	C#
Introduced version	Visual Studio 2017 version 15.3
Values	<p><code>flush_left</code> - Labels are placed at the leftmost column</p> <p><code>one_less_than_current</code> - Labels are placed at one less indent to the current context</p> <p><code>no_change</code> - Labels are placed at the same indent as the current context</p>
Visual Studio default	<code>no_change</code>

Code examples:

```

// csharp_indent_labels= flush_left
class C
{
    private string MyMethod(...)
    {
        if (...) {
            goto error;
        }
    error:
        throw new Exception(...);
    }
}

// csharp_indent_labels = one_less_than_current
class C
{
    private string MyMethod(...)
    {
        if (...) {
            goto error;
        }
    error:
        throw new Exception(...);
    }
}

// csharp_indent_labels= no_change
class C
{
    private string MyMethod(...)
    {
        if (...) {
            goto error;
        }
    error:
        throw new Exception(...);
    }
}

```

csharp_indent_block_contents

Rule name	csharp_indent_block_contents
Applicable languages	C#
Values	<input type="checkbox"/> true - <input type="checkbox"/> false -
Visual Studio default	<input type="checkbox"/> true

Code examples:

```
// csharp_indent_block_contents = true
static void Hello()
{
    Console.WriteLine("Hello");
}

// csharp_indent_block_contents = false
static void Hello()
{
    Console.WriteLine("Hello");
}
```

csharp_indent_braces

Rule name	csharp_indent_braces
Applicable languages	C#
Values	<input type="checkbox"/> true - <input type="checkbox"/> false -
Visual Studio default	<input type="checkbox"/> false

Code examples:

```
// csharp_indent_braces = true
static void Hello()
{
    Console.WriteLine("Hello");
}

// csharp_indent_braces = false
static void Hello()
{
    Console.WriteLine("Hello");
}
```

csharp_indent_case_contents_when_block

Rule name	csharp_indent_case_contents_when_block
Applicable languages	C#
Values	<input type="checkbox"/> true - <input type="checkbox"/> false -
Visual Studio default	<input type="checkbox"/> true

Code examples:

```
// csharp_indent_case_contents_when_block = true
case 0:
{
    Console.WriteLine("Hello");
    break;
}

// csharp_indent_case_contents_when_block = false
case 0:
{
    Console.WriteLine("Hello");
    break;
}
```

Spacing options

These formatting rules concern the use of space characters to format code.

Example `.editorconfig` file:

```
# CSharp formatting settings:
[*.cs]
csharp_space_after_cast = true
csharp_space_after_keywords_in_control_flow_statements = true
csharp_space_between_parentheses = control_flow_statements, type_casts
csharp_space_before_colon_in_inheritance_clause = true
csharp_space_after_colon_in_inheritance_clause = true
csharp_space_around_binary_operators = before_and_after
csharp_space_between_method_declaration_parameter_list_parentheses = true
csharp_space_between_method_declaration_empty_parameter_list_parentheses = false
csharp_space_between_method_declaration_name_and_open_parenthesis = false
csharp_space_between_method_call_parameter_list_parentheses = true
csharp_space_between_method_call_empty_parameter_list_parentheses = false
csharp_space_between_method_call_name_and_opening_parenthesis = false
csharp_space_after_comma = true
csharp_space_before_comma = false
csharp_space_after_dot = false
csharp_space_before_dot = false
csharp_space_after_semicolon_in_for_statement = true
csharp_space_before_semicolon_in_for_statement = false
csharp_space_around_declaration_statements = false
csharp_space_before_open_square_brackets = false
csharp_space_between_empty_square_brackets = false
csharp_space_between_square_brackets = false
```

csharp_space_after_cast

Rule name	csharp_space_after_cast
Applicable languages	C#
Introduced version	Visual Studio 2017 version 15.3
Values	<p><code>true</code> - Place a space character between a cast and the value</p> <p><code>false</code> - Remove space between the cast and the value</p>
Visual Studio default	<code>false</code>

Code examples:

```
// csharp_space_after_cast = true
int y = (int) x;

// csharp_space_after_cast = false
int y = (int)x;
```

csharp_space_after_keywords_in_control_flow_statements

Rule name	csharp_space_after_keywords_in_control_flow_statements
Applicable languages	C#
Introduced version	Visual Studio 2017 version 15.3
Values	<p><code>true</code> - Place a space character after a keyword in a control flow statement such as a <code>for</code> loop</p> <p><code>false</code> - Remove space after a keyword in a control flow statement such as a <code>for</code> loop</p>
Visual Studio default	<code>true</code>

Code examples:

```
// csharp_space_after_keywords_in_control_flow_statements = true
for (int i;i<x;i++) { ... }

// csharp_space_after_keywords_in_control_flow_statements = false
for(int i;i<x;i++) { ... }
```

csharp_space_between_parentheses

Rule name	csharp_space_between_parentheses
Applicable languages	C#
Introduced version	Visual Studio 2017 version 15.3
Values	<p><code>control_flow_statements</code> - Place space between parentheses of control flow statements</p> <p><code>expressions</code> - Place space between parentheses of expressions</p> <p><code>type_casts</code> - Place space between parentheses in type casts</p>
Visual Studio default	<code>false</code>

If you omit this rule or use a value other than `control_flow_statements`, `expressions`, or `type_casts`, the setting is not applied.

Code examples:

```
// csharp_space_between_parentheses = control_flow_statements
for ( int i = 0; i < 10; i++ ) { }

// csharp_space_between_parentheses = expressions
var z = ( x * y ) - ( ( y - x ) * 3 );

// csharp_space_between_parentheses = type_casts
int y = ( int )x;
```

csharp_space_before_colon_in_inheritance_clause

Rule name	csharp_space_before_colon_in_inheritance_clause
Applicable languages	C#
Introduced version	Visual Studio 2017 version 15.7
Values	<p><code>true</code> - Place a space character before the colon for bases or interfaces in a type declaration</p> <p><code>false</code> - Remove space before the colon for bases or interfaces in a type declaration</p>
Visual Studio default	<code>true</code>

Code examples:

```
// csharp_space_before_colon_in_inheritance_clause = true
interface I
{

}

class C : I
{
}

// csharp_space_before_colon_in_inheritance_clause = false
interface I
{

}

class C: I
{}
```

csharp_space_after_colon_in_inheritance_clause

Rule name	csharp_space_after_colon_in_inheritance_clause
Applicable languages	C#
Introduced version	Visual Studio 2017 version 15.7

Values	<input type="checkbox"/> <code>true</code> - Place a space character after the colon for bases or interfaces in a type declaration <input type="checkbox"/> <code>false</code> - Remove space after the colon for bases or interfaces in a type declaration
Visual Studio default	<input type="checkbox"/> <code>true</code>

Code examples:

```
// csharp_space_after_colon_in_inheritance_clause = true
interface I
{
}

class C : I
{

}

// csharp_space_after_colon_in_inheritance_clause = false
interface I
{
}

class C :I
{
```

csharp_space_around_binary_operators

Rule name	csharp_space_around_binary_operators
Applicable languages	C#
Introduced version	Visual Studio 2017 version 15.7
Values	<input type="checkbox"/> <code>before_and_after</code> - Insert space before and after the binary operator <input type="checkbox"/> <code>none</code> - Remove spaces before and after the binary operator <input type="checkbox"/> <code>ignore</code> - Ignore spaces around binary operators
Visual Studio default	<input type="checkbox"/> <code>before_and_after</code>

If you omit this rule, or use a value other than `before_and_after`, `none`, or `ignore`, the setting is not applied.

Code examples:

```
// csharp_space_around_binary_operators = before_and_after
return x * (x - y);

// csharp_space_around_binary_operators = none
return x*(x-y);

// csharp_space_around_binary_operators = ignore
return x * (x-y);
```

csharp_space_between_method_declaration_parameter_list_parentheses

Rule name	csharp_space_between_method_declaration_parameter_list_parentheses
Applicable languages	C#
Introduced version	Visual Studio 2017 version 15.3
Values	<p><code>true</code> - Place a space character after the opening parenthesis and before the closing parenthesis of a method declaration parameter list</p> <p><code>false</code> - Remove space characters after the opening parenthesis and before the closing parenthesis of a method declaration parameter list</p>
Visual Studio default	<code>false</code>

Code examples:

```
// csharp_space_between_method_declaration_parameter_list_parentheses = true
void Bark( int x ) { ... }

// csharp_space_between_method_declaration_parameter_list_parentheses = false
void Bark(int x) { ... }
```

csharp_space_between_method_declaration_empty_parameter_list_parentheses

Rule name	csharp_space_between_method_declaration_empty_parameter_list_parentheses
Applicable languages	C#
Introduced version	Visual Studio 2017 version 15.7
Values	<p><code>true</code> - Insert space within empty parameter list parentheses for a method declaration</p> <p><code>false</code> - Remove space within empty parameter list parentheses for a method declaration</p>
Visual Studio default	<code>false</code>

Code examples:

```
// csharp_space_between_method_declaration_empty_parameter_list_parentheses = true
void Goo( )
{
    Goo(1);
}

void Goo(int x)
{
    Goo();
}

// csharp_space_between_method_declaration_empty_parameter_list_parentheses = false
void Goo()
{
    Goo(1);
}

void Goo(int x)
{
    Goo();
}
```

csharp_space_between_method_declaration_name_and_open_parenthesis

Rule name	csharp_space_between_method_declaration_name_and_open_parenthesis
Applicable languages	C#
Values	<input type="checkbox"/> true - Place a space character between the method name and opening parenthesis in the method declaration <input type="checkbox"/> false - Remove space characters between the method name and opening parenthesis in the method declaration
Visual Studio default	<input type="checkbox"/> false

Code examples:

```
// csharp_space_between_method_declaration_name_and_open_parenthesis = true
void M () { }

// csharp_space_between_method_declaration_name_and_open_parenthesis = false
void M() { }
```

csharp_space_between_method_call_parameter_list_parentheses

Rule name	csharp_space_between_method_call_parameter_list_parentheses
Applicable languages	C#
Introduced version	Visual Studio 2017 version 15.3

Values	<code>true</code> - Place a space character after the opening parenthesis and before the closing parenthesis of a method call <code>false</code> - Remove space characters after the opening parenthesis and before the closing parenthesis of a method call
Visual Studio default	<code>false</code>

Code examples:

```
// csharp_space_between_method_call_parameter_list_parentheses = true
MyMethod( argument );

// csharp_space_between_method_call_parameter_list_parentheses = false
MyMethod(argument);
```

csharp_space_between_method_call_empty_parameter_list_parentheses

Rule name	csharp_space_between_method_call_empty_parameter_list_parentheses
Applicable languages	C#
Introduced version	Visual Studio 2017 version 15.7
Values	<code>true</code> - Insert space within empty argument list parentheses <code>false</code> - Remove space within empty argument list parentheses
Visual Studio default	<code>false</code>

Code examples:

```
// csharp_space_between_method_call_empty_parameter_list_parentheses = true
void Goo()
{
    Goo(1);
}

void Goo(int x)
{
    Goo( );
}

// csharp_space_between_method_call_empty_parameter_list_parentheses = false
void Goo()
{
    Goo(1);
}

void Goo(int x)
{
    Goo();
}
```

csharp_space_between_method_call_name_and_opening_parenthesis

Rule name	csharp_space_between_method_call_name_and_opening_parenthesis
Applicable languages	C#
Introduced version	Visual Studio 2017 version 15.7
Values	<input type="checkbox"/> true - Insert space between method call name and opening parenthesis <input type="checkbox"/> false - Remove space between method call name and opening parenthesis
Visual Studio default	<input type="checkbox"/> false

Code examples:

```
// csharp_space_between_method_call_name_and_opening_parenthesis = true
void Goo()
{
    Goo (1);
}

void Goo(int x)
{
    Goo ();
}

// csharp_space_between_method_call_name_and_opening_parenthesis = false
void Goo()
{
    Goo(1);
}

void Goo(int x)
{
    Goo();
}
```

csharp_space_after_comma

Rule name	csharp_space_after_comma
Applicable languages	C#
Values	<input type="checkbox"/> true - Insert space after a comma <input type="checkbox"/> false - Remove space after a comma
Visual Studio default	<input type="checkbox"/> true

Code examples:

```
// csharp_space_after_comma = true
int[] x = new int[] { 1, 2, 3, 4, 5 };

// csharp_space_after_comma = false
int[] x = new int[] { 1,2,3,4,5 }
```

csharp_space_before_comma

Rule name	csharp_space_before_comma
Applicable languages	C#
Values	<input type="checkbox"/> <code>true</code> - Insert space before a comma <input type="checkbox"/> <code>false</code> - Remove space before a comma
Visual Studio default	<input type="checkbox"/> <code>false</code>

Code examples:

```
// csharp_space_before_comma = true
int[] x = new int[] { 1, 2, 3, 4, 5 };

// csharp_space_before_comma = false
int[] x = new int[] { 1, 2, 3, 4, 5 };
```

csharp_space_after_dot

Rule name	csharp_space_after_dot
Applicable languages	C#
Values	<input type="checkbox"/> <code>true</code> - Insert space after a dot <input type="checkbox"/> <code>false</code> - Remove space after a dot
Visual Studio default	<input type="checkbox"/> <code>false</code>

Code examples:

```
// csharp_space_after_dot = true
this.Goo();

// csharp_space_after_dot = false
this.Goo();
```

csharp_space_before_dot

Rule name	csharp_space_before_dot
Applicable languages	C#

Values	<input type="checkbox"/> true - Insert space before a dot <input type="checkbox"/> false - Remove space before a dot
Visual Studio default	<input type="checkbox"/> false

Code examples:

```
// csharp_space_before_dot = true
this .Goo();

// csharp_space_before_dot = false
this.Goo();
```

csharp_space_after_semicolon_in_for_statement

Rule name	csharp_space_after_semicolon_in_for_statement
Applicable languages	C#
Values	<input type="checkbox"/> true - Insert space after each semicolon in a <code>for</code> statement <input type="checkbox"/> false - Remove space after each semicolon in a <code>for</code> statement
Visual Studio default	<input type="checkbox"/> true

Code examples:

```
// csharp_space_after_semicolon_in_for_statement = true
for (int i = 0; i < x.Length; i++)

// csharp_space_after_semicolon_in_for_statement = false
for (int i = 0;i < x.Length;i++)
```

csharp_space_before_semicolon_in_for_statement

Rule name	csharp_space_before_semicolon_in_for_statement
Applicable languages	C#
Values	<input type="checkbox"/> true - Insert space before each semicolon in a <code>for</code> statement <input type="checkbox"/> false - Remove space before each semicolon in a <code>for</code> statement
Visual Studio default	<input type="checkbox"/> false

Code examples:

```
// csharp_space_before_semicolon_in_for_statement = true
for (int i = 0 ; i < x.Length ; i++)

// csharp_space_before_semicolon_in_for_statement = false
for (int i = 0; i < x.Length; i++)
```

csharp_space_around_declaration_statements

Rule name	csharp_space_around_declaration_statements
Applicable languages	C#
Values	<input type="checkbox"/> ignore - Don't remove extra space characters in declaration statements <input type="checkbox"/> false - Remove extra space characters in declaration statements
Visual Studio default	<input type="checkbox"/> false

Code examples:

```
// csharp_space_around_declaration_statements = ignore
int x = 0;

// csharp_space_around_declaration_statements = false
int x = 0;
```

csharp_space_before_open_square_brackets

Rule name	csharp_space_before_open_square_brackets
Applicable languages	C#
Values	<input type="checkbox"/> true - Insert space before opening square brackets [<input type="checkbox"/> false - Remove space before opening square brackets [
Visual Studio default	<input type="checkbox"/> false

Code examples:

```
// csharp_space_before_open_square_brackets = true
int [] numbers = new int [] { 1, 2, 3, 4, 5 };

// csharp_space_before_open_square_brackets = false
int[] numbers = new int[] { 1, 2, 3, 4, 5 };
```

csharp_space_between_empty_square_brackets

Rule name	csharp_space_between_empty_square_brackets
------------------	--

Applicable languages	C#
Values	<input type="checkbox"/> true - Insert space between empty square brackets [] <input type="checkbox"/> false - Remove space between empty square brackets []
Visual Studio default	<input type="checkbox"/> false

Code examples:

```
// csharp_space_between_empty_square_brackets = true
int[] numbers = new int[ ] { 1, 2, 3, 4, 5 };

// csharp_space_between_empty_square_brackets = false
int[] numbers = new int[] { 1, 2, 3, 4, 5 };
```

csharp_space_between_square_brackets

Rule name	csharp_space_between_square_brackets
Applicable languages	C#
Values	<input type="checkbox"/> true - Insert space characters in non-empty square brackets [0] <input type="checkbox"/> false - Remove space characters in non-empty square brackets [0]
Visual Studio default	<input type="checkbox"/> false

Code examples:

```
// csharp_space_between_square_brackets = true
int index = numbers[ 0 ];

// csharp_space_between_square_brackets = false
int index = numbers[0];
```

Wrap options

These formatting rules concern the use of single lines versus separate lines for statements and code blocks.

Example `.editorconfig` file:

```
# CSharp formatting settings:
[*.cs]
csharp_preserve_single_line_statements = true
csharp_preserve_single_line_blocks = true
```

csharp_preserve_single_line_statements

Rule name	csharp_preserve_single_line_statements
------------------	--

Applicable languages	C#
Introduced version	Visual Studio 2017 version 15.3
Values	<input type="checkbox"/> <code>true</code> - Leave statements and member declarations on the same line <input type="checkbox"/> <code>false</code> - Leave statements and member declarations on different lines
Visual Studio default	<input type="checkbox"/> <code>true</code>

Code examples:

```
//csharp_preserve_single_line_statements = true
int i = 0; string name = "John";

//csharp_preserve_single_line_statements = false
int i = 0;
string name = "John";
```

csharp_preserve_single_line_blocks

Rule name	csharp_preserve_single_line_blocks
Applicable languages	C#
Introduced version	Visual Studio 2017 version 15.3
Values	<input type="checkbox"/> <code>true</code> - Leave code block on single line <input type="checkbox"/> <code>false</code> - Leave code block on separate lines
Visual Studio default	<input type="checkbox"/> <code>true</code>

Code examples:

```
//csharp_preserve_single_line_blocks = true
public int Foo { get; set; }

//csharp_preserve_single_line_blocks = false
public int MyProperty
{
    get; set;
}
```

See also

- [Language conventions](#)
- [Naming conventions](#)
- [.NET coding convention settings for EditorConfig](#)

.NET naming conventions for EditorConfig

10/18/2019 • 7 minutes to read • [Edit Online](#)

Naming conventions concern the naming of code elements such as classes, properties, and methods. For example, you can specify that public members must be capitalized or that asynchronous methods must end in "Async". You can enforce these rules by specifying them in an [.editorconfig file](#). Naming rule violations appear either in the **Error List** or as a suggestion under the name, depending on the severity you choose for your rule. There is no need to build the project in order to see violations.

For each naming convention, you must specify the symbols it applies to, a naming style, and a severity for enforcing the convention, using the properties described below. The order of the properties is not important.

To begin, choose a title for your naming rule that you will use in each of the properties that are needed to fully describe the rule. For example, `public_members_must_be_capitalized` is a good, descriptive name for a naming rule. This page will refer to the title you choose as **<namingRuleTitle>** in the sections that follow.

Symbols

First, identify a group of symbols to apply the naming rule to. This property has the following format:

```
dotnet_naming_rule.<namingRuleTitle>.symbols = <symbolTitle>
```

Give a name to the group of symbols by replacing the **<symbolTitle>** value with a descriptive title, for example `public_symbols`. You'll use the **<symbolTitle>** value in the three property names that describe which symbols the rule is applied to (kinds of symbol, accessibility levels, and modifiers).

Kinds of symbols

To describe the kind of symbols to apply the naming rule to, specify a property in the following format:

```
dotnet_naming_symbols.<symbolTitle>.applicable_kinds = <values>
```

The following list shows the allowable values, and you can specify multiple values by separating them with a comma.

- * (use this value to specify all symbols)
- namespace
- class
- struct
- interface
- enum
- property
- method
- field
- event
- delegate
- parameter
- type_parameter
- local
- local_function

Accessibility levels of symbols

To describe the accessibility levels of the symbols you want the naming rule to apply to, specify a property name in the following format:

```
dotnet_naming_symbols.<symbolTitle>.applicable_accessibilities = <values>
```

The following list shows the allowable values, and you can specify multiple values by separating them with a comma.

- * (use this value to specify all accessibility levels)
- public
- internal or friend
- private
- protected
- protected_internal or protected_friend
- private_protected
- local

The `local` accessibility level applies to symbols defined within a method. It's useful for defining naming conventions for symbols whose accessibility can't be specified in code. For example, if you specify `applicable_accessibilities = local` on a naming convention for constants (`required_modifiers = const`), the rule applies only to constants defined within a method and not those defined in a type.

```
class TypeName
{
    // Constant defined in a type.
    const int X = 3;

    void Method()
    {
        // Constant defined in a method with "local" accessibility.
        const int Y = 4;
    }
}
```

Symbol modifiers (optional)

To describe the modifiers of the symbols you want the naming rule to apply to, specify a property name in the following format:

```
dotnet_naming_symbols.<symbolTitle>.required_modifiers = <values>
```

The following list shows the allowable values (separate multiple values with a comma):

- `abstract` or `must_inherit`
- `async`
- `const`
- `readonly`
- `static` or `shared`

NOTE

If you have a naming rule for `static` or `shared` symbols, it is also applied to `const` symbols because they are implicitly static. If you don't want the `static` naming rule to apply to `const` symbols, create a separate naming rule for `const` symbols.

A naming rule matches signatures that have *all* the modifiers specified in `required_modifiers`. If you omit this property, the default value of an empty list is used, that is, no specific modifiers are required for a match. This means a symbol's modifiers have no effect on whether or not this rule is applied.

TIP

Do not specify a value of `*` for `required_modifiers`. Instead, just omit the `required_modifiers` property altogether and your naming rule will apply to any kind of modifier.

Style

Now that you've identified the group of symbols to apply the naming rule to, you can describe the naming style. A style can be that the name has a certain prefix or a certain suffix, or that individual words in the name are separated with a certain character. You can also specify a capitalization style. The `style` property has the following format:

```
dotnet_naming_rule.<namingRuleTitle>.style = <styleTitle>
```

Give the style a name by replacing the `<styleTitle>` value with a descriptive title, for example `first_word_upper_case_style`. You'll use the `<styleTitle>` value in the property names that describe the naming style (prefix, suffix, word separator character, and capitalization). Use one or more of these properties to describe your style.

Require a prefix

To specify that symbol names must begin with certain characters, use this property:

```
dotnet_naming_style.<styleTitle>.required_prefix = <prefix>
```

Require a suffix

To specify that symbol names must end with certain characters, use this property:

```
dotnet_naming_style.<styleTitle>.required_suffix = <suffix>
```

Require a certain word separator

To specify that individual words in symbol names must be separated with a certain character, use this property:

```
dotnet_naming_style.<styleTitle>.word_separator = <separator character>
```

Require a capitalization style

To specify a particular capitalization style for symbol names, use this property:

```
dotnet_naming_style.<styleTitle>.capitalization = <value>
```

The allowable values for this property are:

- `pascal_case`
- `camel_case`
- `first_word_upper`
- `all_upper`

- all_lower

NOTE

You must specify a capitalization style as part of your naming style, otherwise your naming style might be ignored.

Severity

To describe the severity of a violation of your naming rule, specify a property in the following format:

```
dotnet_naming_rule.<namingRuleTitle>.severity = <value>
```

The following table shows the allowable severity values, and what they mean:

SEVERITY	EFFECT
none	Rule is suppressed completely.
refactoring or silent	When this style is not being followed, do not show anything to the user; however, auto-generated code follows this style.
suggestion	When this style is not being followed, show it to the user as a suggestion, as underlying dots on the first two characters. It has no effect at compile time.
warning	When this style is not being followed, show a compiler warning in the Error List .
error	When this style is not being followed, show a compiler error in the Error List .

NOTE

You do not have to build your project in order to see naming rule violations. They appear as code is edited, either in the **Error List** or as a suggestion.

Rule order

Naming conventions should be ordered from most-specific to least-specific in the EditorConfig file. The first rule encountered that can be applied is the only rule that is applied. However, if there are multiple rule *properties* with the same name, the most recently found property with that name takes precedence. For more information, see [File hierarchy and precedence](#).

Starting in Visual Studio 2019 version 16.2, the order in which naming rules are defined in an EditorConfig file doesn't matter. Instead, Visual Studio orders the naming rules automatically according to the definition of the rules themselves. The [EditorConfig Language Service extension](#) can analyze an EditorConfig file and report cases where the rule ordering in the file is different to what the compiler will use at run time.

If you're using an earlier version of Visual Studio, naming conventions should be ordered from most-specific to least-specific in the EditorConfig file. The first rule encountered that can be applied is the only rule that is applied. However, if there are multiple rule *properties* with the same name, the most recently found property with that name takes precedence. For more information, see [File hierarchy and precedence](#).

Default naming styles

If you don't specify any custom naming rules, Visual Studio uses the following default styles:

- For classes, structs, enumerations, properties, and events with `public`, `private`, `internal`, `protected`, or `protected_internal` accessibility, the default naming style is Pascal case.
- For interfaces with `public`, `private`, `internal`, `protected`, or `protected_internal` accessibility, the default naming style is Pascal case with a required prefix of `I`.

Example

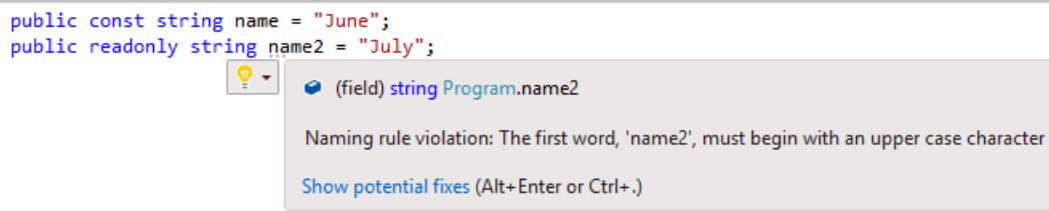
The following `.editorconfig` file contains a naming convention that specifies that public properties, methods, fields, events, and delegates must be capitalized. Notice that this naming convention specifies multiple kinds of symbol to apply the rule to, using a comma to separate the values.

```
# Public members must be capitalized (public_members_must_be_capitalized)
[*.{cs,vb}]
dotnet_naming_rule.public_members_must_be_capitalized.symbols = public_symbols
dotnet_naming_symbols.public_symbols.applicable_kinds = property,method,field,event,delegate
dotnet_naming_symbols.public_symbols.applicable_accessibilities = public
dotnet_naming_symbols.public_symbols.required_modifiers = readonly

dotnet_naming_rule.public_members_must_be_capitalized.style = first_word_upper_case_style
dotnet_naming_style.first_word_upper_case_style.capitalization = first_word_upper

dotnet_naming_rule.public_members_must_be_capitalized.severity = suggestion
```

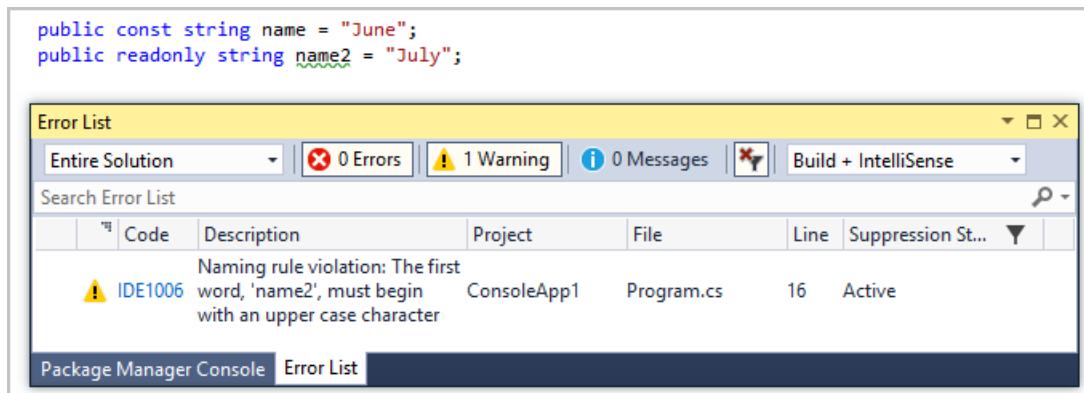
The following screenshot shows the effect of this naming convention in the editor. Two public variables have been named without capitalization of the first letter. One is a `const`, and one is `readonly`. Since the naming rule only applies to `readonly` symbols, only the `readonly` variable shows a naming rule suggestion.



Now let's change the violation severity to `warning`:

```
dotnet_naming_rule.public_members_must_be_capitalized.severity = warning
```

If you close and reopen your code file, instead of seeing the suggestion under the name violation, you see a green squiggle and a warning in the Error List:



See also

- [Language conventions](#)
- [Formatting conventions](#)
- [Roslyn naming conventions](#)
- [Create portable custom editor options](#)
- [.NET coding convention settings for EditorConfig](#)

How to: Customize the scroll bar

10/18/2019 • 2 minutes to read • [Edit Online](#)

When you are working with long code files, it can be hard to keep track of where everything is in the file. You can customize the scroll bar of the code editor to give you an overall picture of what's happening in your code.

Annotations

You can select whether the scroll bar shows annotations such as code changes, breakpoints, bookmarks, errors, and caret position.

1. Open the **Scroll Bars** options page by choosing **Tools > Options > Text Editor > All Languages > Scroll Bars**.
2. Select **Show Annotations over vertical scroll bar**, and then select the annotations you want to see. The available annotations are:

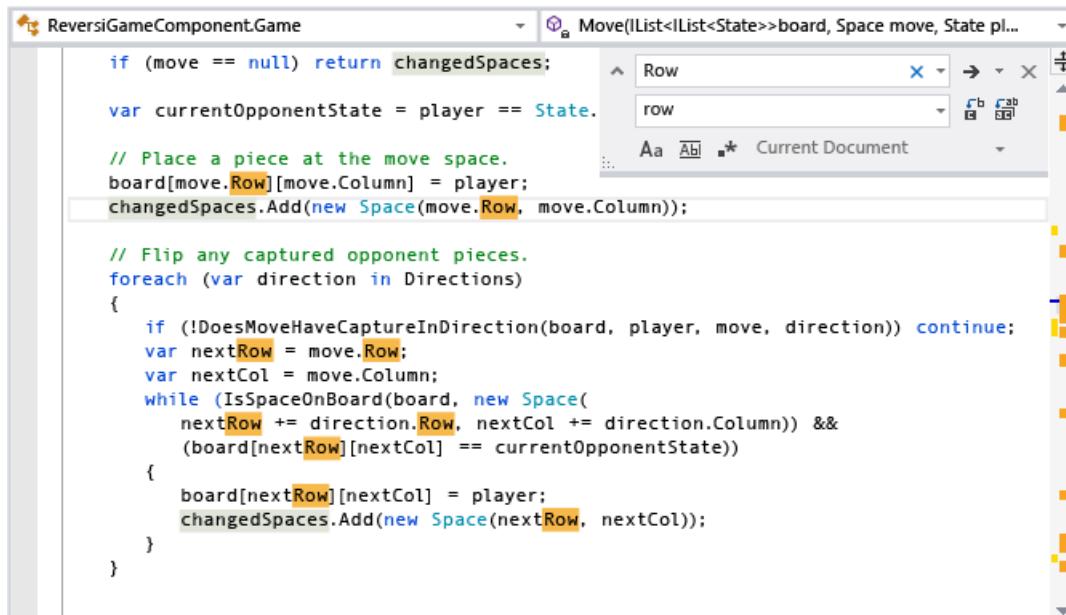
- changes
- marks
- errors
- caret position

TIP

The **Show marks** option includes breakpoints and bookmarks.

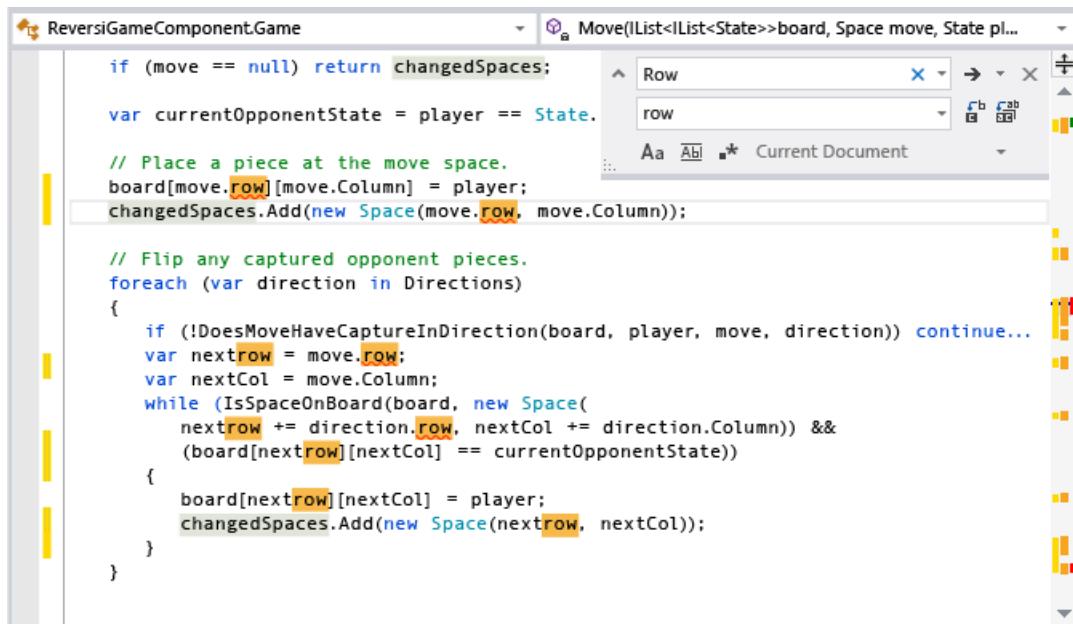
Try it out by opening a large code file and replacing some text that occurs in several places in the file. The scroll bar shows you the effect of the replacements, so you can back out your changes if you replaced something you shouldn't have.

Here's how the scroll bar looks after a search for a string. Notice that all instances of the string appear in the scroll bar.



Here's the scroll bar after replacing all the instances of the string. The red marks in the scroll bar show where the

text replacement introduced errors.



The screenshot shows a code editor window with a scroll bar on the right side. The scroll bar is in 'map mode', which displays a small preview of the code at the current position. The code being edited is a C# method named 'Move'. The scroll bar has a vertical scale from 0 to 100% and a horizontal scale from 0 to 100%. There are several orange highlights on the code, particularly around 'move.row' and 'nextRow' variables, indicating search results or annotations. The status bar at the bottom of the editor shows 'Aa Ab Current Document'.

```
if (move == null) return changedSpaces;

var currentOpponentState = player == State.White ? State.Black : State.White;

// Place a piece at the move space.
board[move.row][move.Column] = player;
changedSpaces.Add(new Space(move.row, move.Column));

// Flip any captured opponent pieces.
foreach (var direction in Directions)
{
    if (!DoesMoveHaveCaptureInDirection(board, player, move, direction)) continue...
    var nextRow = move.row;
    var nextCol = move.Column;
    while (IsSpaceOnBoard(board, new Space(
        nextRow += direction.row, nextCol += direction.Column)) &&
        (board[nextRow][nextCol] == currentOpponentState))
    {
        board[nextRow][nextCol] = player;
        changedSpaces.Add(new Space(nextRow, nextCol));
    }
}
```

Display modes

The scroll bar has two modes: bar mode and map mode.

Bar mode

Bar mode displays annotation indicators on the scroll bar. Clicking on the scroll bar scrolls the page up or down but does not jump to that location in the file.

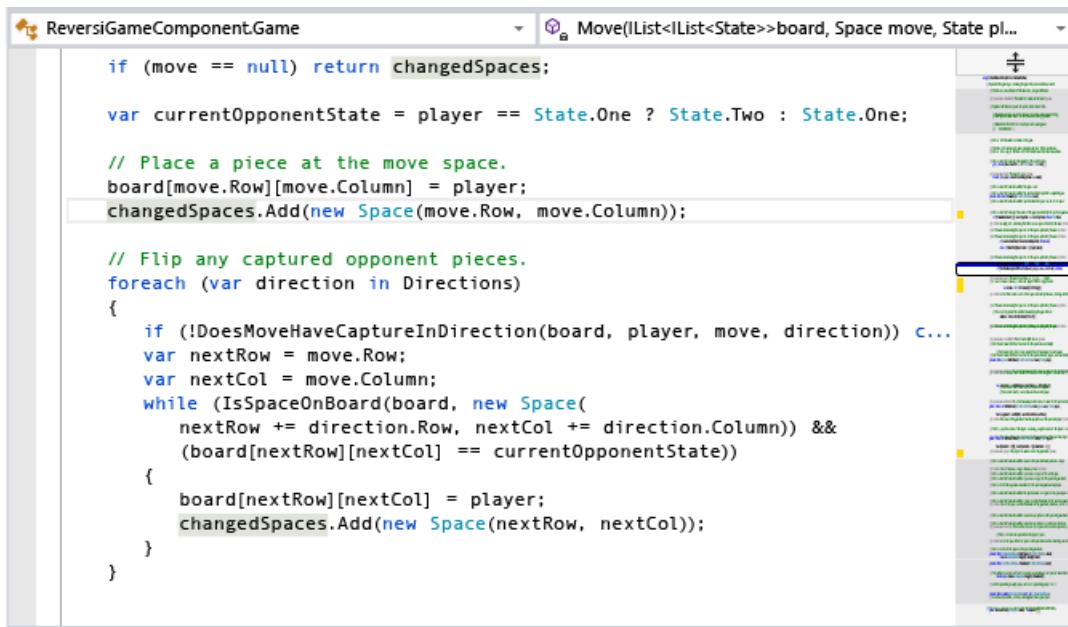
Map mode

In *map mode*, when you click a location on the scroll bar, the cursor jumps to that location in the file instead of just scrolling up or down a page. Lines of code are shown, in miniature, on the scroll bar. You can choose how wide the map column is by selecting a value in **Source overview**. To enable a larger preview of the code when you rest the pointer on the map, select the **Show Preview Tooltip** option. Collapsed regions are shaded differently and expand when you double-click them.

TIP

You can turn the miniature code view off in map mode by setting **Source overview** to **Off**. If **Show Preview Tooltip** is selected, you still see a preview of the code at that location when you hover your pointer on the scroll bar, and the cursor still jumps to that location in the file when you click.

The following image shows the search example when map mode is on and the width is set to **Medium**:



The screenshot shows the Visual Studio code editor with the 'ReversiGameComponent.Game' project open. A tooltip is displayed over the code, specifically over the 'changedSpaces' variable. The tooltip content is: 'Move(IList<IList<State>>board, Space move, State player)'. The 'Show Preview Tooltip' option is checked in the status bar.

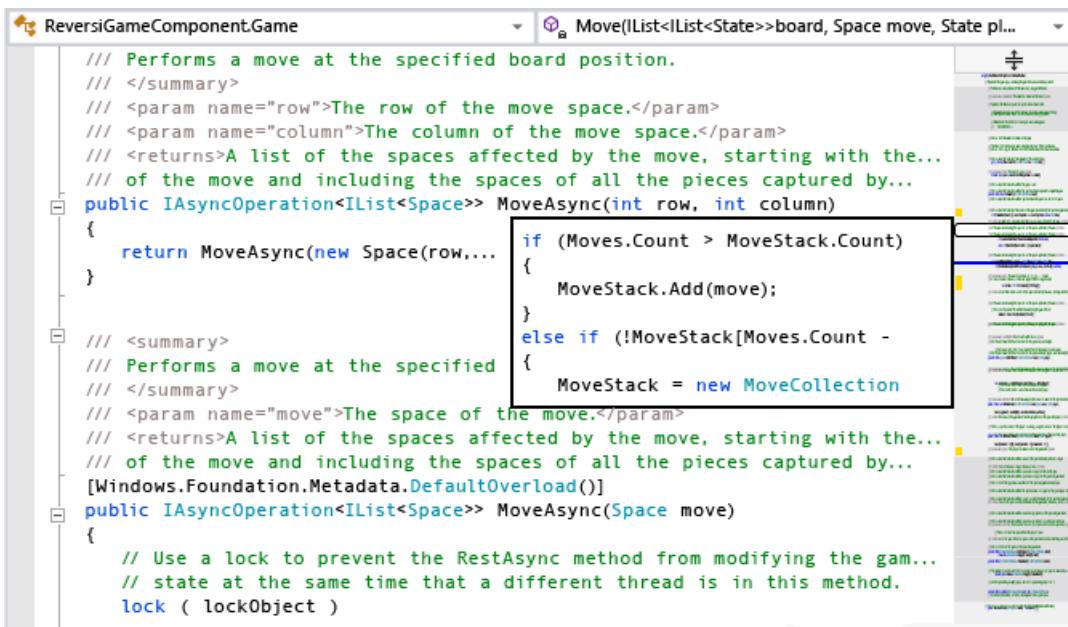
```
if (move == null) return changedSpaces;

var currentOpponentState = player == State.One ? State.Two : State.One;

// Place a piece at the move space.
board[move.Row][move.Column] = player;
changedSpaces.Add(new Space(move.Row, move.Column));

// Flip any captured opponent pieces.
foreach (var direction in Directions)
{
    if (!DoesMoveHaveCaptureInDirection(board, player, move, direction)) ...
    var nextRow = move.Row;
    var nextCol = move.Column;
    while (IsSpaceOnBoard(board, new Space(
        nextRow += direction.Row, nextCol += direction.Column)) &&
        (board[nextRow][nextCol] == currentOpponentState))
    {
        board[nextRow][nextCol] = player;
        changedSpaces.Add(new Space(nextRow, nextCol));
    }
}
```

The following image shows the **Show Preview Tooltip** option:



The screenshot shows the Visual Studio code editor with the 'ReversiGameComponent.Game' project open. A tooltip is displayed over the 'MoveAsync' method signature. The tooltip content is: 'MoveAsync(int row, int column)'. The 'Show Preview Tooltip' option is checked in the status bar.

```
/// Performs a move at the specified board position.
/// </summary>
/// <param name="row">The row of the move space.</param>
/// <param name="column">The column of the move space.</param>
/// <returns>A list of the spaces affected by the move, starting with the...
/// of the move and including the spaces of all the pieces captured by...
public IAsyncOperation<IList<Space>> MoveAsync(int row, int column)
{
    return MoveAsync(new Space(row, ...
}

/// <summary>
/// Performs a move at the specified
/// </summary>
/// <param name="move">The space of the move.</param>
/// <returns>A list of the spaces affected by the move, starting with the...
/// of the move and including the spaces of all the pieces captured by...
[Windows.Foundation.Metadata.DefaultOverload()]
public IAsyncOperation<IList<Space>> MoveAsync(Space move)
{
    // Use a lock to prevent the RestAsync method from modifying the gam...
    // state at the same time that a different thread is in this method.
    lock ( lockObject )
}
```

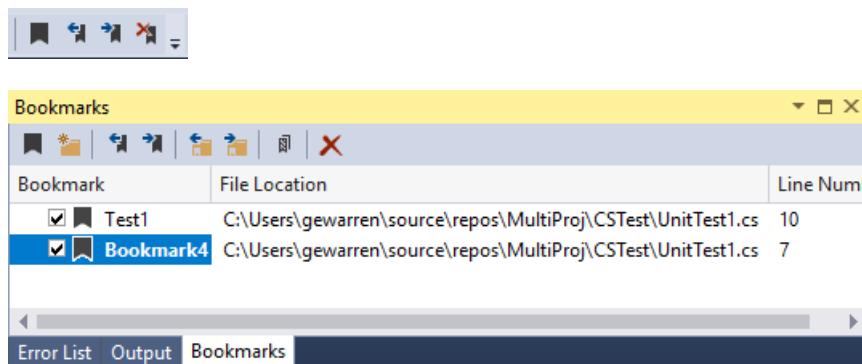
See also

- [Features of the code editor](#)

Set bookmarks in code

10/18/2019 • 2 minutes to read • [Edit Online](#)

You can use bookmarks to mark lines in your code so that you can quickly return to a specific location, or jump back and forth between locations. Bookmark commands and icons are available in two places: the **Bookmark Window** (**View > Bookmark Window**) and the text editor toolbar.



Manage bookmarks

To add a bookmark, place the cursor on the line you want to bookmark. Choose the **Toggle a bookmark** button, or press **Ctrl+K, Ctrl+K**. This adds the bookmark. If you choose the **Toggle a bookmark** button (or press **Ctrl+K, Ctrl+K**) again, the bookmark is removed.

To know at a glance what a specific bookmark is for, you can rename it in the **Bookmark Window** from the right-click or context menu. You can delete bookmarks by choosing the **Delete** button in the bookmark window.

IMPORTANT

The bookmark is set to the line number, not to the code. If you modify the code, the bookmark is retained at the line number, and does not move with the code.

You can navigate between bookmarks by using the **Next bookmark** and **Previous bookmark** buttons in the bookmark window.

You can organize bookmarks into virtual folders by choosing **New Folder** in the bookmark window and then dragging selected bookmarks into the new folder.

You can turn off bookmarks (without removing them) by choosing the **Disable All Bookmarks** button in the bookmark window. You can re-enable them by choosing the same button (which is now called **Enable All Bookmarks**).

See also

- [Features of the code editor](#)

Find code changes and other history with CodeLens

11/26/2019 • 9 minutes to read • [Edit Online](#)

CodeLens lets you stay focused on your work while you find out what happened to your code—without leaving the editor. You can find references to a piece of code, changes to your code, linked bugs, work items, code reviews, and unit tests.

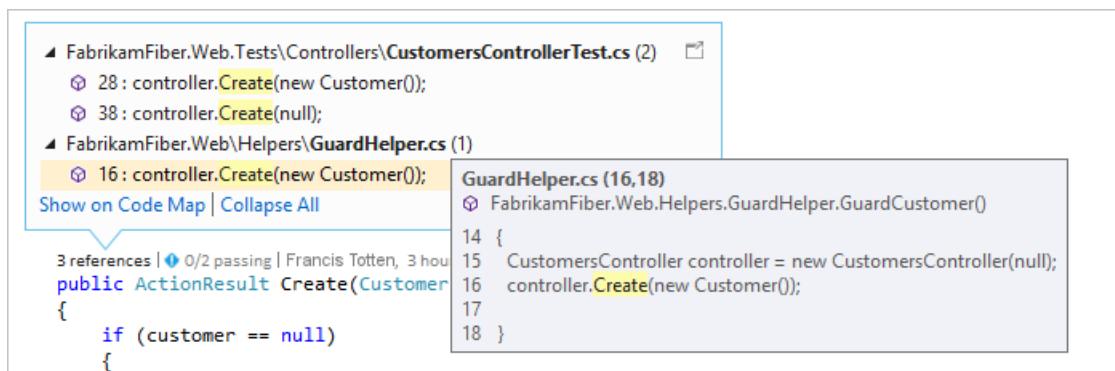
NOTE

CodeLens is available in Visual Studio Community edition, however, the *source control* indicators are not available in this edition.

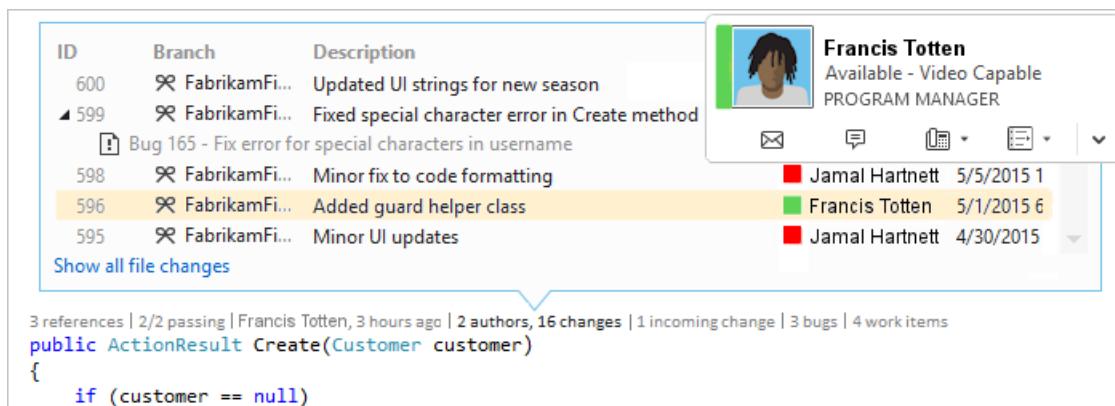
NOTE

CodeLens is available only in Visual Studio Enterprise and Professional editions. It is not available in Visual Studio Community edition.

See where and how the individual parts of your code are used in your solution:



Contact your team about changes to your code without leaving the editor:



To choose the indicators that you want to see, or to turn CodeLens off and on, go to **Tools > Options > Text Editor > All Languages > CodeLens**.

Find references to your code

You can find references in C# or Visual Basic code.

1. Choose the **references** indicator or press **Alt+2**.

The screenshot shows the Visual Studio Code Map tool interface. At the top, there are two entries in the list:

- FabrikamFiber.Web.Tests\Controllers\CustomersControllerTest.cs (2)**
 - 28 : controller.Create(new Customer());
 - 38 : controller.Create(null);
- FabrikamFiber.Web\Helpers\GuardHelper.cs (1)**
 - 16 : controller.Create(new Customer());

Below the list, a code snippet for the `Create` method is shown:

```
3 references | 0/2 passing | Francis Totten, 3 hours ago | 2 authors, 16 changes | 1 incoming change | 3 bugs | 4 work items
public ActionResult Create(Customer customer)
{
    if (customer == null)
    {
```

NOTE

If the indicator shows **0 references**, you have no references from C# or Visual Basic code. However, there may be references in other items such as `.xaml` and `.aspx` files.

2. To view the referencing code, mouse over the reference in the list.

The screenshot shows the Visual Studio Code Map tool interface with a detailed view of the `ParseTerm()` method in `Parser.cs`.

The method signature is:

```
14 |
15 CustomersController controller = new CustomersController(null);
16 controller.Create(new Customer());
17 |
18 }
```

A callout points to the line `controller.Create(new Customer());` which is highlighted with a yellow box. A tooltip for this reference shows:

GuardHelper.cs (16,18)
↳ FabrikamFiber.Web.Helpers.GuardHelper.GuardCustomer()

3. To open the file that contains the reference, double-click the reference.

Code maps

To see relationships between the code and its references, [create a code map](#). In the code map shortcut menu, select **Show All References**.

The screenshot shows Microsoft Visual Studio with the Code Map feature open. On the left, the code editor shows the `Parser.cs` file with the `ParseTerm()` method selected. On the right, the `CodeMap1.dgml` window displays a flowchart of references:

```
graph TD
    PE[ParseExpression] --> PT[ParseTerm]
```

The `ParseTerm()` node is highlighted with a yellow box.

Find changes in your code

Inspect your code's history to find out what happened to your code. Or, review changes before they're merged into your code so you can better understand how changes in other branches might affect your code.

You need:

- Visual Studio Enterprise or Professional edition
- Azure DevOps Services, Team Foundation Server 2013 or later, or Git
- [Skype for Business](#) to contact your team from the code editor

For C# or Visual Basic code that's stored with Team Foundation Version Control (TFVC) or Git, you get CodeLens details at the class and method levels (*code element-level* indicators). If your Git repository is hosted in TfGit, you also get links to TFS work items.



For file types other than .cs or .vb, you get CodeLens details for the entire file in one place at the bottom of the window (*file-level* indicators).



Code element-level indicators

Code element-level indicators let you see who changed your code and what changes they made. Code element-level indicators are available for C# and Visual Basic code.

This is what you see when you use Team Foundation Version Control (TFVC) in Team Foundation Server or Azure DevOps Services:

ID	Branch	Description	Author	Date
600	FabrikamFi...	Updated UI strings for new season	Francis Totten	5/21/2015
599	FabrikamFi...	Fixed special character error in Create method	Francis Totten	5/18/2015
		Bug 165 - Fix error for special characters in username		
598	FabrikamFi...	Minor fix to code formatting	Jamal Hartnett	5/5/2015 1
596	FabrikamFi...	Added guard helper class	Francis Totten	5/1/2015 6
595	FabrikamFi...	Minor UI updates	Jamal Hartnett	4/30/2015

Show all file changes

3 references | 2/2 passing | Francis Totten, 3 hours ago 2 authors, 16 changes | 1 incoming change | 3 bugs | 4 work items

```
public ActionResult Create(Customer customer)
{
    if (customer == null)
```

The default time period is the last 12 months. If your code is stored in Team Foundation Server, you can change

the time period by running the [TFSConfig command](#) with the [CodeIndex command](#) and the **/indexHistoryPeriod** flag.

To see a detailed history of all the changes, including those from more than a year ago, choose **Show all file changes**:

The screenshot shows a list of code changes in a Git repository. At the bottom of the list, there is a button labeled "Show all file changes". Below this button, a tooltip displays the file content:

```
public ActionResult Create(Customer customer)
{
    if (customer == null)
```

The **History** window opens:

The History window for `CustomersController.cs` shows a list of changesets. The first changeset, ID 600, is selected. A tooltip below the list displays the file content:

```
public ActionResult Create(Customer customer)
{
    if (customer == null)
```

When your files are in a Git repository and you choose the code element-level changes indicator, this is what you see:

The History window for `CustomersController.cs` in a Git repository shows a list of changesets. The first changeset, ID 9350d639, is selected. A tooltip below the list displays the file content:

```
public ActionResult Create(Customer customer)
{
    //check model state
    if (ModelState.IsValid)
    {
        this.customerRepository.InsertOrUpdate(customer);
    }
}
```

The tooltip also displays the following details for the selected changeset:

- ID: 9350d639
- Date: 5/18/2015 4:32:16 AM
- Author: Francis Totten (francist@fabrikam.com)
- Change type: Edit
- Description: Fixed special characters error in Create method
- Related Work Items: #165

File-level indicators

Find changes for an entire file in the file-level indicators at the bottom of the window:

The screenshot shows the Visual Studio IDE with the Team Explorer tab selected. In the center, there's a list of commits from a Git repository. The commits are:

- 9350d639 Fixed special characters error in Create method... Francis Totten (francist@cr... 5/18/2015
 - 165 - Fix error for special characters in username
- f0c985ea Minor UI presentation adjustments Jamal Hartnett (jamal@fa... 4/30/2015
- ea00c408 Initial Commit Mateo Escobedo (mateo@... 4/17/2015

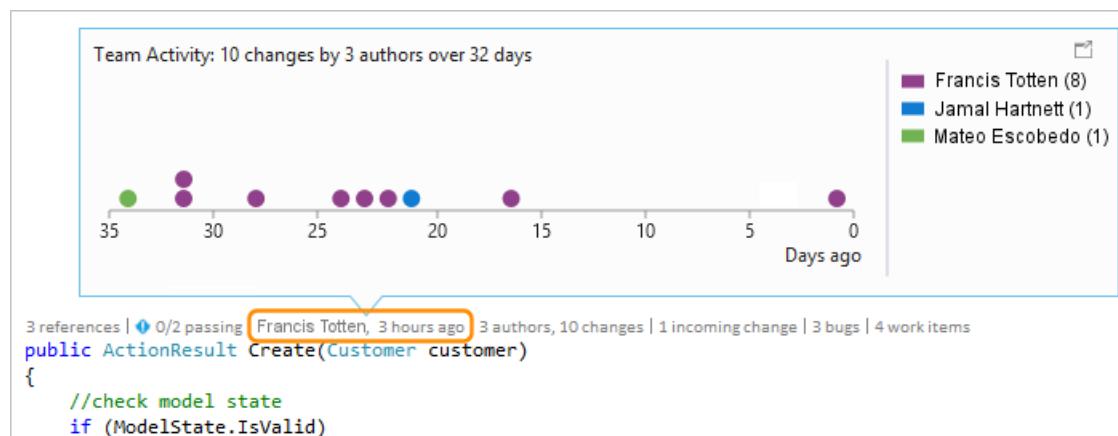
At the bottom of the commit list, there's a status bar with the text "3 authors, 3 changes".

NOTE

File-level indicators are not available for C# and Visual Basic files.

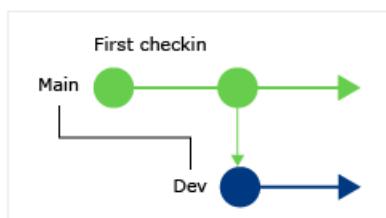
To get more details about a change, right-click that item. Depending on whether you are using TFVC or Git, there are options to compare the versions of the file, view details and track the changeset, get the selected version of the file, and email the author of that change. Some of these details appear in **Team Explorer**.

You can also see who changed your code over time. This can help you find patterns in your team's changes and assess their impact.



Find changes in your current branch

Your team may have multiple branches, for example a main branch and a child development branch, to reduce the risk of breaking stable code.



You can find out how many people changed your code and how many changes were made in the main branch by pressing **Alt+6**:

The screenshot shows the Visual Studio IDE with the Team Explorer tab selected. A specific commit is highlighted in the list:

ID	Branch	Description	Author	Date
13	FabrikamFiber.CallCenter-Main	Initial checkin	Francis Totten	12/17/2013

Below the commit list, there's a status bar with the text "3 references | 2/2 passing | Francis Totten, 3 hours ago | 2 authors, 16 changes | 1 incoming change | 3 bugs | 4 work items". The code diff for the commit is shown at the bottom:

```

public static void ThrowIfNullOrEmpty(string value, string name)
{
    if (string.IsNullOrEmpty(value))
        throw new ArgumentNullException(name);
}
  
```

Find when your code was branched

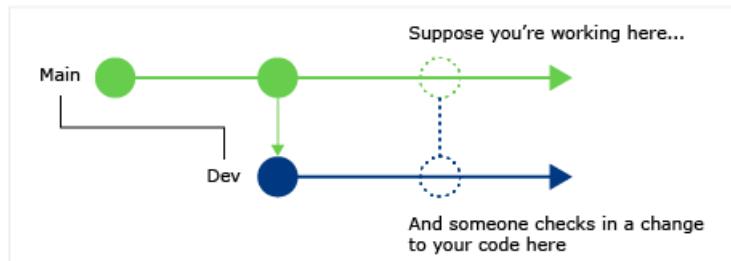
To find when your code was branched, navigate to your code in the child branch. Then, select the **changes** indicator or press **Alt+6**:

ID	Branch	Description	Author	Date
▲ 15	FabrikamFiber.CallCenter-Dev (from...)	Branched from Main	Franci...	12/17/2...
▷ 13	FabrikamFiber.CallCenter-Main	First checkin	Franci...	12/17/2...

3 references | 2/2 passing | Francis Totten, 3 hours ago | 2 authors, 16 changes | 1 incoming change | 3 bugs | 4 work items

```
public static void ThrowIfNullOrEmpty(string value, string name)
{
    if (string.IsNullOrEmpty(value))
        throw new ArgumentNullException(name);
```

Find incoming changes from other branches



You can view incoming changes. In the following screenshot, a bug fix was made in the "Dev" branch:

ID	Branch	Description	Author	Date
▲ 17	FabrikamFiber.CallCenter-Dev	Fix bug for null name	Francis T...	12/17/2013
▷ 15	FabrikamFiber.CallCenter-Dev	Branched from Main	Francis T...	12/17/2013

3 references | 2/2 passing | Francis Totten, 3 hours ago | 2 authors, 16 changes | 1 incoming change | 3 bugs | 4 work items

```
public static void ThrowIfNullOrEmpty(string value, string name)
{
    if (string.IsNullOrEmpty(value))
        throw new ArgumentNullException(name);
```

You can review the change without leaving your current branch ("Main"):

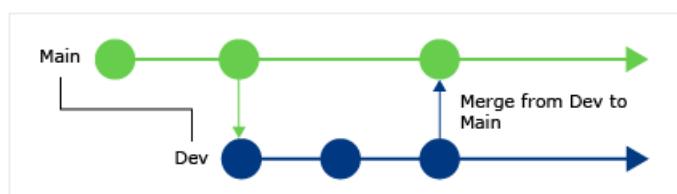
ID	Branch	Description	Author	Date
▲ 17	FabrikamFiber.CallCenter-Dev	Fix bug for null name	Francis Totten	12/17/2013
▷ 13	FabrikamFiber.CallCenter-Main	First checkin	Francis Totten	12/17/2013

3 references | 2/2 passing | Francis Totten, 3 hours ago | 2 authors, 16 changes | 1 incoming change | 3 bugs | 4 work items

```
public static void ThrowIfNullOrEmpty(string value, string name)
{
    if (string.IsNullOrEmpty(value))
        throw new ArgumentNullException(name);
```

Find when changes got merged

You can see when changes got merged, so you can determine which changes are included in your branch:



For example, your code in the Main branch now has the bug fix from the "Dev" branch:

ID Branch Description Author

- 18 FabrikamFiber.CallCenter-Main (from Fabri... Merge from Dev to Main Local Version Fra...
- 17 FabrikamFiber.CallCenter-Dev Fix bug for null name Fra...
- 13 FabrikamFiber.CallCenter-Main Fix checkin Fra...

3 references | 2/2 passing | Francis Totten, 3 hours ago | 2 authors, 16 changes | 1 incoming change | 3 bugs | 4 work items

```
public static void ThrowIfNullOrEmpty(string value, string name)
{
    if (string.IsNullOrEmpty(value))
        throw new ArgumentNullException(name);
```

Compare an incoming change with your local version

Compare an incoming change with your local version by pressing **Shift+F10**, or by double-clicking the changeset.

ID Branch Description Author

- 600 FabrikamFi... Updated UI strings for new season Local Version Fra...
- 599 FabrikamFi... Fixed special character error in Create method Fra...
- 598 FabrikamFi... Minor fix to code formatting Fra...
- 596 FabrikamFi... Added guard helper class Fra...
- 595 FabrikamFi... Minor UI updates Fra...
- 594 FabrikamFi... Minor fix to formatting Fra...

Show all file changes

3 references | 2/2 passing | Francis Totten, 3 hours ago | 2 authors, 16 changes | 1 incoming change

```
public ActionResult Create(Customer customer)
{
    if (customer == null)
```

Compare with Local Version

- Changeset Details
- Track Changeset 600
- Hide Intermediate Branch Merges
- Send IM to Francis Totten
- Call Francis Totten
- Video Chat with Francis Totten
- Send Email to Francis Totten
- Open Contact Card

Branch icons

The icon in the **Branch** column tells you how the branch is related to the branch you're working in.

ICON	THE CHANGE CAME FROM:
🔗	The current branch
🕒	The parent branch
🕒	A child branch
🕒	A peer branch
🕒	A branch further away than a parent, child, or peer
🔗	A merge from the parent branch to a child branch
🔗	A merge from a child branch to the parent branch
🔗	A merge from an unrelated branch (baseless merge)

Linked work items

Find linked work items by selecting the **work items** indicator or by pressing **Alt+8**.

Task 158
Francis Totten (francist@fabrikam.com)
Update to support category 2 accounts
4/20/2015 2:48:31 AM

This item was last associated with a changeset in \$/FabrikamFiber/FabrikamFiber.CallCenter-Dev, the current branch.

ID	Type	Branch	Title
▲ 164	Task	Fabrik...	Review performance against new data sto
		Description 593 - FabrikamFiber.CallCenter-Dev - Perf review with r	
▲ 162	Task	Fabrik...	Refactor for dependency injection of auth
		Description 590 - FabrikamFiber.CallCenter-Dev - Implemented dep	for auth
▲ 158	Task	Fabrik...	Update to support category 2 accounts
		Description 582 - FabrikamFiber.CallCenter-Dev - Update to support	Francis Totten (fran... 4/20/2015)
93	Product	Fabrik...	Application to support Fabrikam Call cent
		Francis Totten (fran... 12/16/2013)	▼

3 references | 2/2 passing | Francis Totten, 3 hours ago | 2 authors, 16 changes | 1 incoming change | 3 bugs | 4 work items

```
public ActionResult Create(Customer customer)
{
    if (customer == null)
```

Linked code reviews

Find linked code reviews by selecting the **reviews** indicator. To use the keyboard, hold down the **Alt** key and then press **Left arrow** or **Right arrow** to navigate the indicator options.

ID	Type	Title	Author	Date
► 32	Code Review Request	More comments for Calculator	Jamal Hartnett	8/4/2013
► 26	Code Review Request	Added comments for Calculator	Jamal Hartnett	8/2/2013

3 references | 2/2 passing | Francis Totten, 3 hours ago | 2 authors, 16 changes | 1 incoming change | 3 bugs | 4 work items | 2 reviews

```
public ActionResult Create(Customer customer)
{
    if (customer == null)
```

Linked bugs

Find linked bugs by selecting the **bugs** indicator or by pressing **Alt+7**.

ID	Type	Branch	Title	Changeset Author Date
▲ 165	Bug	Fabrik...	Fix error for special characters in username	Francis Totten (fra 5/21/2015)
		Description 599 - FabrikamFiber.CallCenter-Dev - Fixed special char:		
▲ 160	Bug	Fabrik...	Changes to repository code require addition	Jamal Hartnett (ja 4/20/2015)
		Description 583 - FabrikamFiber.CallCenter-Dev - Updated to match		
▲ 154	Bug	Fabrik...	UT failures for customer create	Francis Totten (fra 4/16/2015)
		Description 579 - FabrikamFiber.CallCenter-Release - Fix test		

ten, 3 hours ago | 2 authors, 16 changes | 1 incoming change | 3 bugs | 4 work items

```
public ActionResult Create(Customer customer)
{
    if (customer == null)
        throw new NullReferenceException();
}
```

Description 599
Francis Totten (francist@fabrikam.com)
Fixed special character error in Create method
This changeset was made in \$/FabrikamFiber/FabrikamFiber.CallCenter-Dev, the current branch.

Contact the owner of an item

Find the author of an item by selecting the **authors** indicator or by pressing **Alt+5**.

The screenshot shows the Visual Studio CodeLens interface. On the right, there is a contact card for Francis Totten, which includes a photo, name, status (Available - Video Capable), title (PROGRAM MANAGER), and a list of recent interactions with Jamal Hartnett (5/5/2015 1, 5/1/2015 6, 4/30/2015). Below the contact card, there are icons for messaging, calling, calendar, and task list. At the bottom of the card, there is a dropdown menu.

ID **Branch** **Description**

- 600 FabrikamFi... Updated UI strings for new season
- 599 FabrikamFi... Fixed special character error in Create method
 - Bug 165 - Fix error for special characters in username
 - Minor fix to code formatting
 - Added guard helper class
 - Minor UI updates
- 598 FabrikamFi... Minor fix to code formatting
- 596 FabrikamFi... Added guard helper class
- 595 FabrikamFi... Minor UI updates

Show all file changes

3 references | 2/2 passing | Francis Totten, 3 hours ago | 2 authors, 16 changes | 1 incoming change | 3 bugs | 4 work items

```
public ActionResult Create(Customer customer)
{
    if (customer == null)
```

Open the shortcut menu for an item to see the contact options. If you have Lync or Skype for Business installed, you see these options:

The screenshot shows the Visual Studio CodeLens interface with a context menu open over a changeset item. The menu includes options like 'Compare with Local Version', 'Changeset Details', 'Hide Intermediate Branch Merges' (which is checked), 'Send IM to Francis Totten', 'Call Francis Totten', 'Video Chat with Francis Totten', 'Send Email to Francis Totten', and 'Open Contact Card'. The menu is highlighted with an orange border.

ID **Branch** **Description**

- 600 FabrikamFi... Updated UI strings for new season
- 599 FabrikamFi... Fixed special character error in Create method
- 598 FabrikamFi... Minor fix to code formatting
- 596 FabrikamFi... Added guard helper class
- 595 FabrikamFi... Minor UI updates
- 594 FabrikamFi... Minor fix to formatting

Show all file changes

3 references | 2/2 passing | Francis Totten, 3 hours ago | 2 authors, 16 changes | 1 incoming change

```
public ActionResult Create(Customer customer)
{
    if (customer == null)
```

Associated unit tests

You can discover unit tests that exist for your C# or Visual Basic code without opening **Test Explorer**.

1. Go to application code that has associated [unit test code](#).
2. If you have not already, build your application to load the CodeLens test indicators.
3. Review the tests for the code by pressing **Alt+3**.

The screenshot shows the Visual Studio CodeLens interface with a tooltip displaying unit test results. The tooltip has two sections: 'Test' and 'Duration'. The 'Test' section lists two tests: 'CreateInsertsCustomerAndSaves' (failed) and 'CreateNullCustomer' (passed). The 'Duration' section shows '854 ms' for the failed test and '11 ms' for the passed test. At the bottom of the tooltip, there are 'Run All' and 'Run' buttons.

Test

- CreateInsertsCustomerAndSaves
- CreateNullCustomer

Duration

- 854 ms
- 11 ms

Run All | Run

3 references | 1/2 passing | Francis Totten, 3 hours ago | 2 authors, 16 changes | 1 incoming change | 3 bugs | 4 work items

```
public ActionResult Create(Customer customer)
{
    if (customer == null)
```

4. If you see a warning icon , the tests haven't run yet, so run them.

The screenshot shows the Visual Studio CodeLens interface with a tooltip displaying unit test results. The tooltip has two sections: 'Test' and 'Duration'. The 'Test' section lists two tests: 'CreateInsertsCustomerAndSaves' (failed) and 'CreateNullCustomer' (passed). The 'Duration' section shows '854 ms' for the failed test and '11 ms' for the passed test. At the bottom of the tooltip, there are 'Run All' and 'Run' buttons.

Test

- CreateInsertsCustomerAndSaves
- CreateNullCustomer

Duration

- 854 ms
- 11 ms

Run All | Run

3 references | 0/2 passing | Francis Totten, 3 hours ago | 2 authors, 16 changes | 1 incoming change | 3 bugs | 4 work items

```
public ActionResult Create(Customer customer)
{
    if (customer == null)
```

5. To review a test's definition, double-click the test item in the CodeLens indicator window to open the code

file in the editor.

```
[TestClass()]
0 references | Jia Hao Tseng | 1 author, 1 change | 1 work item
public class CustomersControllerTest
{
    MockCustomerRepository mockCustomerRepo;
    CustomersController controller;

    [TestInitialize()]
    0 references | Jia Hao Tseng | 1 author, 1 change | 1 work item
    public void SetupController()
    {
        mockCustomerRepo = new MockCustomerRepository();
        controller = new CustomersController(mockCustomerRepo);
    }

    [TestMethod()]
    0 | 0 references | Jia Hao Tseng | 1 author, 1 change | 1 work item
    public void CreateInsertsCustomerAndSaves()
    {
        controller.Create(new Customer());

        Assert.IsTrue(mockCustomerRepo.IsInsertOrUpdateCalled);
        Assert.IsTrue(mockCustomerRepo.IsSaveCalled);
    }

    [TestMethod()]
    [ExpectedException(typeof(ArgumentNullException))]
    0 | 0 references | Jia Hao Tseng | 1 author, 1 change | 1 work item
    public void CreateNullCustomer()
    {
        controller.Create(null);
    }
}
```

6. To review the test's results, choose the test status indicator (✖ or ✓) or press **Alt+1**.

✖ Test Failed - CreateInsertsCustomerAndSaves

Message: `Assert.IsTrue` failed.

Elapsed time: 854 ms

.StackTrace:

```
CustomersControllerTest.CreateInsertsCustomerAndSaves()
```

Run | Debug

testMethod()

✖ 0 references | Jia Hao Tseng | 1 author, 1 change | 1 work item

```
public void CreateInsertsCustomerAndSaves()
{
    controller.Create(new Customer());

    Assert.IsTrue(mockCustomerRepo.IsInsertOrUpdateCalled);
    Assert.IsTrue(mockCustomerRepo.IsSaveCalled);
}
```

7. To see how many people changed this test, who changed this test, or how many changes were made to this test, [find your code's history](#) and linked items.

Keyboard shortcuts

To use the keyboard to select indicators, press and hold the **Alt** key to display the related number keys, then press the number that corresponds to the indicator you want to select.

2 3 4 5 6 7 8

3 references | 0/2 passing | Francis Totten, 3 hours ago | 2 authors, 16 changes | 1 incoming change | 3 bugs | 4 work items

```
public ActionResult Create(Customer customer)
{
    if (customer == null)
```

NOTE

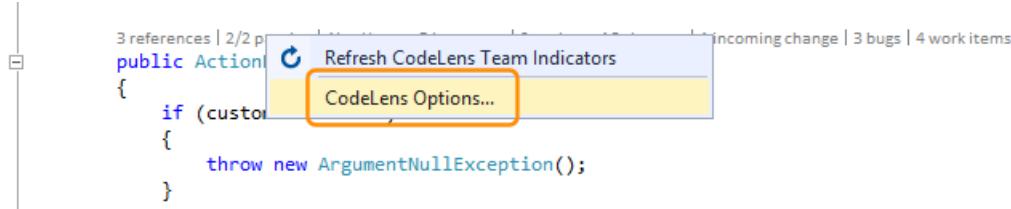
To select the **reviews** indicator, hold down **Alt** while using the left and right arrow keys to navigate.

Q & A

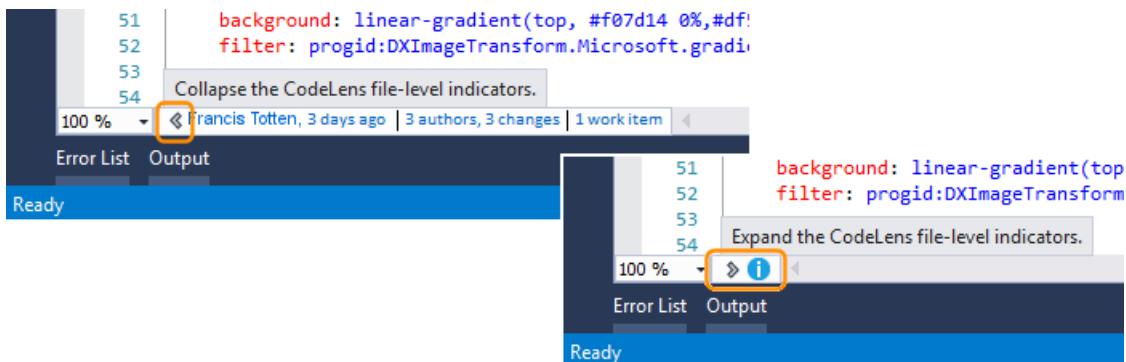
Q: How do I turn CodeLens off or on or choose which indicators to see?

A: You can turn indicators off or on, except for the references indicator. Go to **Tools > Options > Text Editor > All Languages > CodeLens**.

When the indicators are turned on, you can also open the CodeLens options from the indicators.



Turn CodeLens file-level indicators on and off using the chevron icons at the bottom of the editor window.



Q: Where is CodeLens?

A: CodeLens appears in C# and Visual Basic code at the method, class, indexer, and property level. CodeLens appears at the file level for all other types of files.

- Make sure CodeLens is turned on. Go to **Tools > Options > Text Editor > All Languages > CodeLens**.
- If your code is stored in TFS, make sure that code indexing is turned on by using the **CodeIndex** command with the [TFS Config command](#).
- DevOps-related indicators appear only when work items are linked to the code and when you have permissions to open linked work items. Confirm that you have [team member permissions](#).
- Unit test indicators don't appear when application code doesn't have unit tests. Test status indicators appear automatically in test projects. If you know that your application code has unit tests, but the test indicators don't appear, try building the solution (**Ctrl+Shift+B**).

TIP

CodeLens is available in Visual Studio Community edition, however, the *source control* indicators are not available in this edition.

TIP

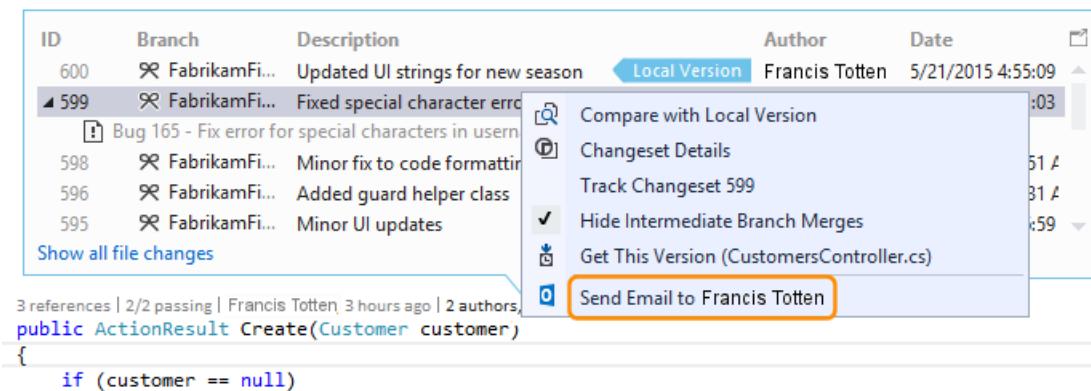
CodeLens is not available in Visual Studio Community edition.

Q: Why don't I see the work item details for a commit?

A: This might happen because CodeLens can't find the work items in Azure Boards or TFS. Check that you're connected to the project that has those work items, and that you have permissions to see those work items. Work item details might also not show if the commit description has incorrect information about the work item IDs in Azure Boards or TFS.

Q: Why don't I see the Skype indicators?

A: Skype indicators don't appear if you're not signed into Skype for Business, don't have it installed, or don't have a supported configuration. However, you can still send email:

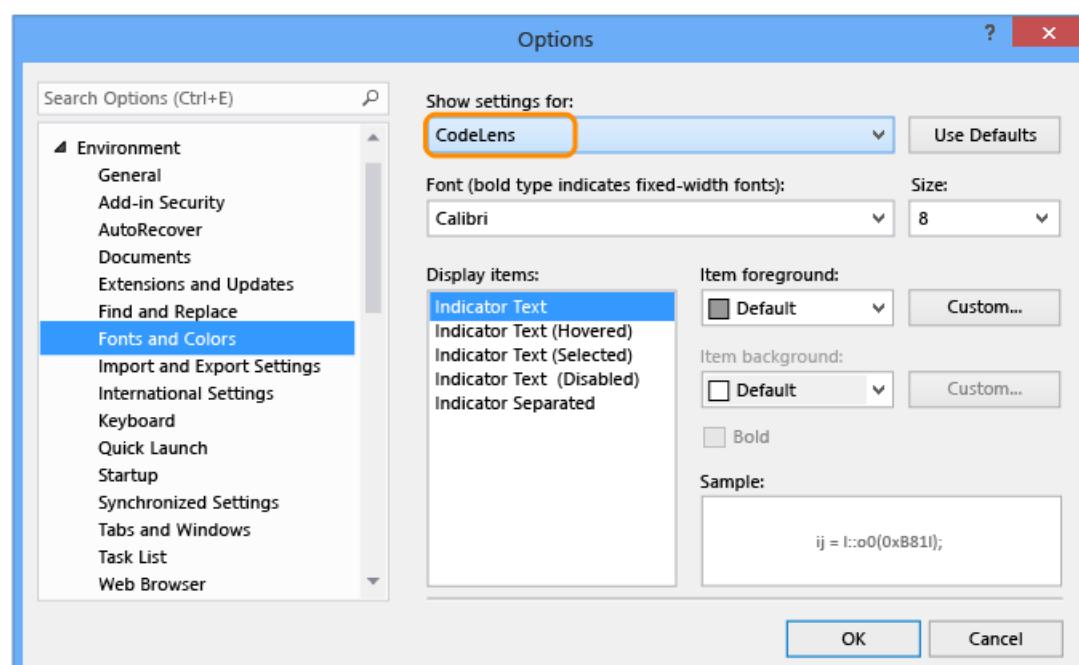
**Which Skype and Lync configurations are supported?**

- Skype for Business (32-bit or 64-bit)
- Lync 2010 or later alone (32-bit or 64-bit), but not Lync Basic 2013 with Windows 8.1

CodeLens doesn't support having different versions of Lync or Skype installed. They might not be localized for all localized versions of Visual Studio.

Q: How do I change the font and color for CodeLens?

A: Go to **Tools > Options > Environment > Fonts and Colors**.

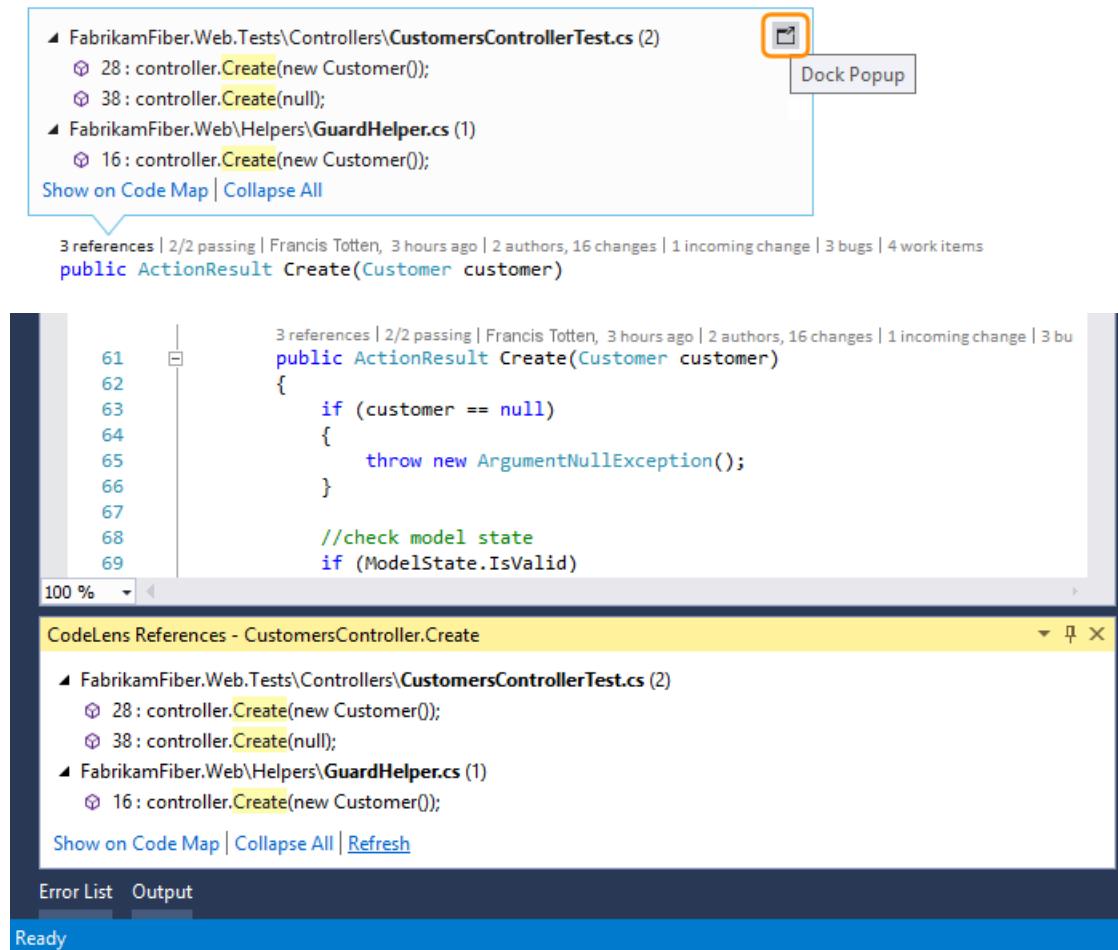


To use the keyboard:

1. Press **Alt+T+O** to open the **Options** dialog box.
2. Press **Up Arrow** or **Down Arrow** to go to the **Environment** node, then press **Left Arrow** to expand the node.
3. Press **Down Arrow** to go to **Fonts and Colors**.
4. Press **Tab** to go to the **Show settings for** list, and then press **Down Arrow** to select **CodeLens**.

Q: Can I move the CodeLens heads-up display?

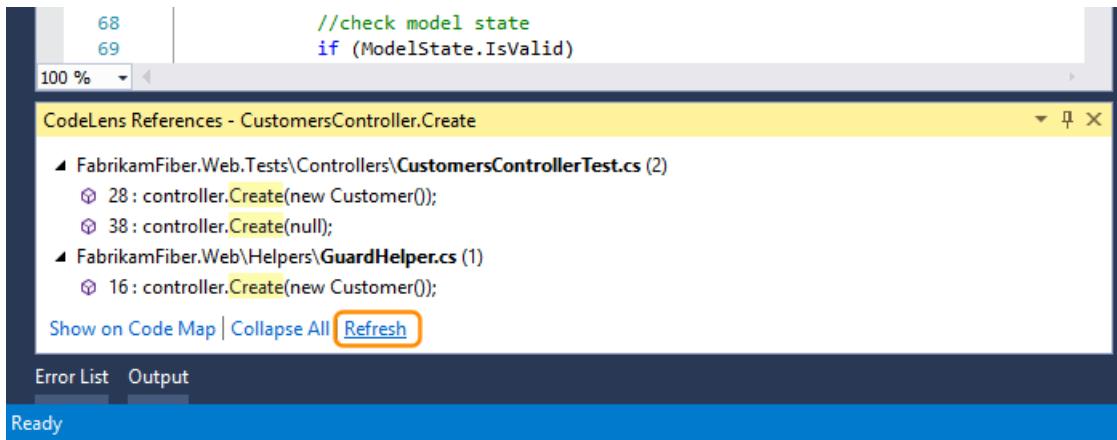
A: Yes, choose  to dock CodeLens as a window.



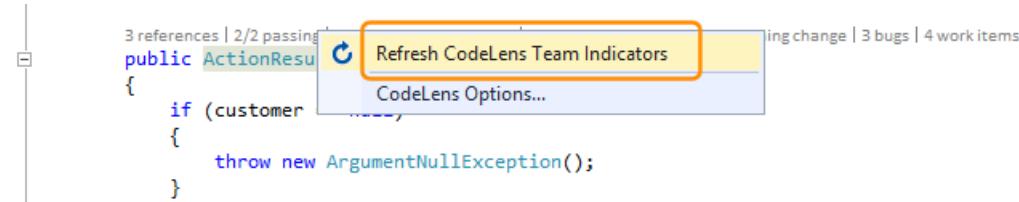
Q: How do I refresh the indicators?

A: This depends on the indicator:

- **References:** This indicator updates automatically when the code changes. If the **References** indicator is docked as a separate window, refresh the indicator by selecting **Refresh**:



- **Team:** Refresh these indicators by selecting **Refresh CodeLens Team Indicators** from the right-click menu:



- **Test:** Find unit tests for your code to refresh the **Test** indicator.

Q: What's "Local Version"?

A: The **Local Version** arrow points at the most recent changeset in your local version of a file. When the server has more recent changesets, they appear above or below the **Local Version** arrow, depending on the order used to sort the changesets.

Q: Can I manage how CodeLens processes code to show history and linked items?

A: Yes. If your code is in TFS, use the [CodeIndex command](#) with the [TFS Config command](#).

Q: My CodeLens test indicators no longer appear in my file when I first open my solution. How can I load them?

A: Rebuild your project to get CodeLens test indicators to load in your file. To improve performance, Visual Studio no longer fetches source information for test indicators when code files are loaded. Test indicators are loaded after a build, or when you navigate to a test by double-clicking on it in [Test Explorer](#).

See also

- [Features of the code editor](#)

CodeIndex command

11/26/2019 • 3 minutes to read • [Edit Online](#)

Use the **CodeIndex** command to manage code indexing on Team Foundation Server. For example, you might want to reset the index to fix CodeLens information, or turn off indexing to investigate server performance issues.

Required permissions

To use the **CodeIndex** command, you must be a member of the **Team Foundation Administrators** security group. See [Permissions and groups defined for Azure DevOps Services and TFS](#).

NOTE

Even if you log on with administrative credentials, you must open an elevated Command Prompt window to run this command. You must also run this command from the application tier for Team Foundation.

Syntax

```
TFSCore CodeIndex /indexingStatus | /setIndexing:[ on | off | keepupOnly ] | /ignoreList:[ add | remove |  
removeAll | view ] ServerPath | /listLargeFiles [/fileCount:FileCount] [/minSize:MinSize] | /reindexAll |  
/destroyCodeIndex [/noPrompt] | /temporaryDataSizeLimit:[ view | <SizeInGBs> | disable ] |  
/indexHistoryPeriod:[ view | all | <NumberOfMonths> ] [/collectionName:CollectionName |  
/collectionId:CollectionId]
```

Parameters

ARGUMENT	DESCRIPTION
<code>CollectionName</code>	Specifies the name of the project collection. If the name has spaces, enclose the name with quotation marks, for example, "Fabrikam Website".
<code>CollectionId</code>	Specifies the identification number of the project collection.
<code>ServerPath</code>	Specifies the path to a code file.

OPTION	DESCRIPTION
<code>/indexingStatus</code>	Show the status and configuration of the code indexing service.
<code>/setIndexing:[on off keepupOnly]</code>	<ul style="list-style-type: none">- on: Start indexing all changesets.- off: Stop indexing all changesets.- keepupOnly: Stop indexing previously created changesets and start indexing new changesets only.

OPTION	DESCRIPTION
/ignoreList: [add remove removeAll view] <input type="text" value="ServerPath"/> <p>You can use the wildcard character (*) at the start, end, or both ends of the server path.</p>	<p>Specifies a list of code files and their paths that you don't want indexed.</p> <ul style="list-style-type: none"> - add: Add the file that you don't want indexed to the ignored file list. - remove: Remove the file that you want indexed from the ignored file list. - removeAll: Clear the ignored file list and start indexing all files. - view: See all the files that aren't being indexed.
/listLargeFiles [/fileCount: <input type="text" value="FileCount"/> /minSize: <input]<="" td="" type="text" value="MinSize"/> <td data-bbox="782 518 1447 653"> Shows the specified number of files that exceeds the specified size in KB. You can then use the /ignoreList option to exclude these files from indexing. </td>	Shows the specified number of files that exceeds the specified size in KB. You can then use the /ignoreList option to exclude these files from indexing.
/reindexAll	Clear previously indexed data and restart indexing.
/destroyCodeIndex [/noPrompt]	Delete the code index and remove all indexed data. Does not require confirmation if you use the /noPrompt option.
/temporaryDataSizeLimit: [view < <input type="text" value="SizeInGBs"/> > disable]	Control how much temporary data that CodeLens creates when processing changesets. The default limit is 2 GB. <ul style="list-style-type: none"> - view: Show the current size limit. - <input type="text" value="SizeInGBs"/> : Change the size limit. - disable: Remove the size limit. <p>This limit is checked before CodeLens processes a new changeset. If temporary data exceeds this limit, CodeLens will pause processing past changesets, not new ones. CodeLens will restart processing after the data is cleaned up and falls below this limit. Cleanup runs automatically once a day. This means temporary data might exceed this limit until cleanup starts running.</p>
/indexHistoryPeriod: [view all < <input type="text" value="NumberOfMonths"/> >]	Control how long to index your change history. This affects how much history CodeLens shows you. The default limit is 12 months. This means CodeLens shows your change history from the last 12 months only. <ul style="list-style-type: none"> - view: Show the current number of months. - all: Index all change history. - <input type="text" value="NumberOfMonths"/> : Change the number of months used to index change history.
/collectionName: <input type="text" value="CollectionName"/>	Specifies the name of the project collection on which to run the CodeIndex command. Required if you don't use /CollectionId .
/collectionId: <input type="text" value="CollectionId"/>	Specifies the identification number of the project collection on which to run the CodeIndex command. Required if you don't use /CollectionName .

Examples

NOTE

The example companies, organizations, products, domain names, email addresses, logos, people, places, and events depicted herein are fictitious. No association with any real company, organization, product, domain name, email address, logo, person, places, or events is intended or should be inferred.

To see the code indexing status and configuration:

```
TFSCfg Config CodeIndex /indexingStatus /collectionName:"Fabrikam Website"
```

To start indexing all changesets:

```
TFSCfg Config CodeIndex /setIndexing:on /collectionName:"Fabrikam Website"
```

To stop indexing previously created changesets and start indexing new changesets only:

```
TFSCfg Config CodeIndex /setIndexing:keepupOnly /collectionName:"Fabrikam Website"
```

To find up to 50 files that are larger than 10 KB:

```
TFSCfg Config CodeIndex /listLargeFiles /fileCount:50 /minSize:10 /collectionName:"Fabrikam Website"
```

To exclude a specific file from indexing and add it to the ignored file list:

```
TFSCfg Config CodeIndex /ignoreList:add "$/Fabrikam Website/Catalog.cs" /collectionName:"Fabrikam Website"
```

To see all the files that aren't indexed:

```
TFSCfg Config CodeIndex /ignoreList:view
```

To clear previously indexed data and restart indexing:

```
TFSCfg Config CodeIndex /reindexAll /collectionName:"Fabrikam Website"
```

To save all changeset history:

```
TFSCfg Config CodeIndex /indexHistoryPeriod:all /collectionName:"Fabrikam Website"
```

To remove the size limit on CodeLens temporary data and continue indexing regardless of temporary data size:

```
TFSCfg Config CodeIndex /temporaryDataSizeLimit:disable /collectionName:"Fabrikam Website"
```

To delete the code index with confirmation:

```
TFSCfg Config CodeIndex /destroyCodeIndex /collectionName:"Fabrikam Website"
```

See also

- [Find code changes and other history with CodeLens](#)
- [Managing server configuration with TFS Config](#)

Add Visual Studio editor support for other languages

10/18/2019 • 2 minutes to read • [Edit Online](#)

Learn about how the Visual Studio editor supports reading and navigating through different computer languages and how you can add Visual Studio editor support for other languages.

Syntax colorization, statement completion, and Navigate To support

Features in the Visual Studio editor such as syntax colorization, statement completion (also known as IntelliSense), and *Navigate To* can help you more easily write, read, and edit your code. The following screenshot shows an example of editing a Perl script in Visual Studio. The syntax is automatically colorized. For example, remarks in the code are colored green, code is black, paths are red, and statements are blue. The Visual Studio editor automatically applies syntax colorization to any language it supports. In addition, when you begin to enter a known language keyword or object, statement completion displays a list of possible statements and objects. Statement completion can help you write code more quickly and easily.

```
81 if ($ENV{'HTTP_USER_AGENT'} =~ /scooter|robot|ultraseek|spider|wget/i) {
82     return;
83     # En: Don't Display Adverts for robots.
84     # Fr: N'affiche pas la pub pour les robots.
85 }
86
87
88 if ($ENV{'HTTP_USER_AGENT'} =~ /Win/i) { $OSVisitor = "1" }
89 elsif ($ENV{'HTTP_USER_AGENT'} =~ /Macintosh/i) { $OSVisitor = "2" }
90 elsif ($ENV{'HTTP_USER_AGENT'} =~ /Linux/i) { $OSVisitor = "3" }
91 elsif ($ENV{'HTTP_USER_AGENT'} =~ /X11/i) { $OSVisitor = "4" }
92 elsif ($ENV{'HTTP_USER_AGENT'} =~ /scooter|robot|ultraseek|spider|wget/i) { $OSVisitor = "5" }
93 else { $OSVisitor = "6" }
94
95 $1
96 $Lang
97 $LangVisitor
98 $Lg
99 $LgAndKw
100 $Line
101 $LogFile
102 if (&TestLang("De", $QueryLang, $Host, $Lang, $ScriptName)) {
103     $LangVisitor = "De";
104 } elsif (&TestLang("Es", $QueryLang, $Host, $Lang, $ScriptName)) {
105     $LangVisitor = "Es";
106 } else {
107     $LangVisitor = "En";
108 }
```

Visual Studio currently provides syntax colorization and basic statement completion support for the following languages using [TextMate Grammars](#). If your favorite language isn't in the table, though, don't worry—you can add it.

Bat	F#	Java	Markdown	Rust	Visual Basic
Clojure	Go	JavaDoc	Objective-C	ShaderLab	C#
CMake	Groovy	JSON	Perl	ShellScript	Visual C++
CoffeeScript	HTML	LESS	Python	SQL	VBNet

CSS	INI	LUA	R	Swift	XML
Docker	Jade	Make	Ruby	TypeScript	YAML

In addition to syntax colorization and basic statement completion, Visual Studio also has a feature called [Navigate To](#). This feature enables you to quickly search code files, file paths, and code symbols. Visual Studio provides Navigate To support for the following languages.

- C#
- C++
- TypeScript
- JavaScript
- Visual Basic
- Go
- Java
- PHP

All of these file types have the features described earlier even if support for a given language hasn't yet been installed. Installing specialized support for some languages may provide additional language support, such as IntelliSense or other advanced language features like light bulbs.

Add support for non-supported languages

Visual Studio provides language support in the editor by using [TextMate Grammars](#). If your favorite programming language currently isn't supported in the Visual Studio editor, first, search the web—a TextMate bundle for the language may already exist. If you can't find one, though, you can add support for it yourself by creating a TextMate bundle model for language grammars and snippets.

Add any new TextMate Grammars for Visual Studio in the following folder:

`%userprofile%\vs\Extensions`

Under this base path, add the following folders if they apply to your situation:

FOLDER NAME	DESCRIPTION
<code>\<language name></code>	The language folder. Replace <code><language name></code> with the name of the language. For example, <code>\Matlab</code> .
<code>\Syntaxes</code>	The grammar folder. Contains the grammar.json files for the language, such as <code>Matlab.json</code> .
<code>\Snippets</code>	The snippets folder. Contains snippets for the language.

In Windows, `%userprofile%` resolves to the path: `c:\Users\<user name>`. If the `Extensions` folder does not exist on your system, you will need to create it. If the folder already exists, it will be hidden.

TIP

If you have any files open in the editor, you'll need to close and reopen them to see syntax highlighting after you add the TextMate Grammars.

For details about how to create TextMate Grammars, see [TextMate - Introduction to Language Grammars](#) and [Notes on how to create a Language Grammar and Custom Theme for a Textmate Bundle](#).

See also

- [Add a Language Server Protocol extension](#)
- [Walkthrough: Create a code snippet](#)
- [Walkthrough: Display statement completion](#)

View the structure of code using different tool windows

10/18/2019 • 10 minutes to read • [Edit Online](#)

You can examine classes and their members in Visual Studio using various tool windows, including **Class View**, **Call Hierarchy**, **Object Browser**, and **Code Definition** (C++ only). These tool windows can examine code in Visual Studio projects, .NET components, COM components, dynamic-link libraries (DLL), and type libraries (TLB).

You can also use **Solution Explorer** to browse the types and members in your projects, search for symbols, view a method's call hierarchy, find symbol references, and more, without having to switch between multiple tool windows.

If you have Visual Studio Enterprise edition, you can use *code maps* to visualize the structure of your code and its dependencies across the entire solution. For more information, see [Map dependencies with code maps](#).

Class View (Visual Basic, C#, C++)

Class View is shown as part of **Solution Explorer** and as a separate window. **Class View** displays the elements of an application. The upper pane displays namespaces, types, interfaces, enumerations, and classes, and the lower pane displays the members that belong to the type selected in the upper pane. By using this window, you can move to member definitions in the source code (or in the **Object Browser** if the element is defined outside your solution).

You do not have to compile a project to view its elements in **Class View**. The window is refreshed as you modify the code in your project.

You can add code to your project by selecting the project node and choosing the **Add** button to open the **Add New Item** dialog box. The code is added in a separate file.

If your project is checked in to source code control, every **Class View** element displays an icon that indicates the source code status of the file. Common source code control commands such as **Check Out**, **Check In**, and **Get Latest Version** are also available on the shortcut menu for the element.

Class View toolbar

The **Class View** toolbar contains the following commands:

New Folder	Creates a virtual folder or subfolder in which you can organize frequently used elements. They are saved in the active solution (.suo) file. After you rename or delete an element in your code, it might appear in a virtual folder as an error node. To correct this problem, delete the error node. If you renamed an element, you can move it from the project hierarchy into the folder again.
Back	Navigates to the previously selected item.
Forward	Navigates to the next selected item.

View Class Diagram (managed code projects only)	Becomes available when you select a namespace or type in Class View . When a namespace is selected, the class diagram shows all the types in it. When a type is selected, the class diagram shows only that type.
--	--

Class View settings

The **Class View Settings** button on the toolbar has the following settings:

Show Base Types	Base types are displayed.
Show Project References	Project references are displayed.
Show Hidden Types and Members	Hidden types and members (not intended for use by clients) are displayed in light gray text.
Show Public Members	Public members are displayed.
Show Protected Members	Protected members are displayed.
Show Private Members	Private members are displayed.
Show Other Members	Other kinds of members are displayed, including internal (or Friend in Visual Basic) members.
Show Inherited Members	Inherited members are displayed.

Class View shortcut menu

The shortcut (or right-click) menu in **Class View** may contain the following commands, depending on the kind of project selected:

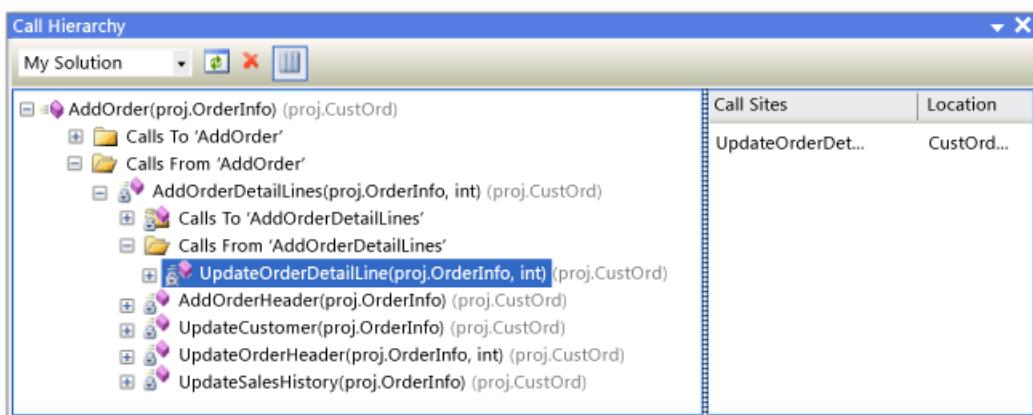
Go To Definition	Finds the definition of the element in the source code, or in the Object Browser , if the element is not defined in the open project.
Browse Definition	Displays the selected item in the Object Browser .
Find All References	Finds the currently selected object item and displays the results in a Find Results window.
Filter To Type (managed code only)	Displays only the selected type or namespace. You can remove the filter by choosing the Clear Find (X) button next to the Find box.
Copy	Copies the fully qualified name of the item.
Sort Alphabetically	Lists types and members alphabetically by name.
Sort by Member Type	Lists types and members in order by type (such that classes precede interfaces, interfaces precede delegates, and methods precede properties).

Sort by Member Access	Lists types and members in order by access type, such as public or private.
Group by Member Type	Sorts types and members into groups by object type.
Go To Declaration (C++ code only)	Displays the declaration of the type or member in the source code, if available.
Go To Definition	Displays the definition of the type or member in the source code, if available.
Go To Reference	Displays a reference to the type or member in the source code, if available.
View Call Hierarchy	Displays the selected method in the Call Hierarchy window.

Call Hierarchy window (Visual Basic, C#, C++)

The **Call Hierarchy** window shows where a given method or property is called. It also lists the methods that are called from that method. You can view multiple levels of the call graph, which shows the caller-callee relationships among the methods in a specified scope.

You can display the **Call Hierarchy** window by selecting a method (or property or constructor) in the editor and then choosing **View Call Hierarchy** on the shortcut menu. The display should resemble the following image:



By using the drop-down list on the toolbar, you can specify the scope of the hierarchy: the solution, current project, or current document.

The main pane displays the calls to and from the method, and the **Call Sites** pane displays the location of the selected call. For members that are virtual or abstract, an **Overrides method name** node appears. For interface members, an **Implements method name** node appears.

The **Call Hierarchy** window does not find method group references, which include places where a method is added as an event handler or is assigned to a delegate. To find these references, use the **Find All References** command.

The shortcut menu in the **Call Hierarchy** window contains the following commands:

Add as New Root	Adds the selected node as a new root node.
------------------------	--

Remove Root	Removes the selected root node from the tree view pane.
Go to Definition	Navigates to the original definition of a method.
Find All References	Finds in the project all the references to the selected method.
Copy	Copies the selected node (but not its subnodes).
Refresh	Refreshes the information.

Object Browser

The **Object Browser** window displays descriptions of the code in your projects.

You can filter the components you want to view by using the drop-down list at the top of the window. Custom components can include managed code executables, library assemblies, type libraries, and .ocx files. It is not possible to add C++ custom components.

Custom settings are saved in the Visual Studio user application directory,
%APPDATA%\Microsoft\VisualStudio\15.0\ObjBrowEX.dat.

Custom settings are saved in the Visual Studio user application directory,
%APPDATA%\Microsoft\VisualStudio\16.0\ObjBrowEX.dat.

The left pane of the **Object Browser** shows assemblies. You can expand the assemblies to display the namespaces they contain, and then expand the namespaces to display the types they contain. When you select a type, its members (such as properties and methods) are listed in the right pane. The lower right pane displays detailed information about the selected item.

You can search for a specific item by using the **Search** box at the top of the window. Searches are case-insensitive. Search results are displayed in the left pane. To clear a search, choose the **Clear Search (X)** button next to the **Search** box.

The **Object Browser** keeps track of the selections you have made, and you can navigate among your selections by using the **Forward** and **Back** buttons on the toolbar.

You can use the **Object Browser** to add an assembly reference to an open solution by selecting an item (assembly, namespace, type, or member) and choosing the **Add Reference** button on the toolbar.

Object Browser settings

By using the **Object Browser Settings** button on the toolbar, you can specify one of the following views:

View Namespaces	Displays namespaces rather than physical containers, in the left pane. Namespaces stored in multiple physical containers are merged.
View Containers	Displays physical containers rather than namespaces, in the left pane. View Namespaces and View Containers are mutually exclusive settings.
Show Base Types	Displays base types.

Show Hidden Types and Members	Displays hidden types and members (not intended for use by clients), in light gray text.
Show Public Members	Displays public members.
Show Protected Members	Displays protected members.
Show Private Members	Displays private members.
Show Other Members	Displays other types of members, including internal (or Friend in Visual Basic) members.
Show Inherited Members	Displays inherited members.
Show Extension Methods	Displays extension methods.

Object Browser shortcut menu commands

The shortcut (or right-click) menu in **Object Browser** may contain the following commands, depending on the kind of item selected:

Browse Definition	Shows the primary node for the selected item.
Find All References	Finds the currently selected object item and displays the results in a Find Results window.
Filter To Type	Displays only the selected type or namespace. You can remove the filter by choosing the Clear Search button.
Copy	Copies the fully qualified name of the item.
Remove	If the scope is a custom component set, removes the selected component from the scope.
Sort Alphabetically	Lists types and members alphabetically by name.
Sort by Object Type	Lists types and members in order by type (such that classes precede interfaces, interfaces precede delegates, and methods precede properties).
Sort by Object Access	Lists types and members in order by access type, such as public or private.
Group by Object Type	Sorts types and members into groups by object type.
Go To Declaration (C++ projects only)	Displays the declaration of the type or member in the source code, if available.
Go To Definition	Displays the definition of the type or member in the source code, if available.

Go To Reference	Displays a reference to the type or member in the source code, if available.
View Call Hierarchy	Displays the selected method in the Call Hierarchy window.

Code Definition window (C++)

The **Code Definition** window displays the definition of a selected C++ type or member in the active project. The type or member can be selected in the code editor or in a code view window.

Although this window is read-only, you can set breakpoints or bookmarks in it. To modify the displayed definition, choose **Edit Definition** on the shortcut menu. This opens the source file in the code editor and moves the insertion point to the line where the definition begins.

NOTE

Starting in Visual Studio 2015, the **Code Definition** window can only be used with C++ code.

Code Definition shortcut menu

The shortcut (or right-click) menu in the **Code Definition** window may contain the following commands:

Quick Actions and Refactorings	
Rename	
Generate Graph of Include Files	
Peek Definition	
Go To Definition	Finds the definition (or definitions, for partial classes) and displays them in a Find Results window.
Go To Declaration	
Find All References	Finds the references to the type or member in the solution.
View Call Hierarchy	Displays the method in the Call Hierarchy window.
Toggle Header / Code File	
Run Tests	If there are unit tests in the project, runs the tests for the selected code.
Debug Tests	
Breakpoint	Inserts a breakpoint (or a tracepoint).
Run to Cursor	Runs the program in debug mode to the location of the cursor.

Snippet	
Cut, Copy, Paste	
Annotation	
Outlining	Standard outlining commands.
Rescan	
Edit Definition	Moves the insertion point to the definition in the code window.
Choose Encoding	Opens the Encoding window so that you can set an encoding for the file.

Document Outline window

You can use the **Document Outline** window in conjunction with designer views, such as the designer for a XAML page or a Windows Form designer, or with HTML pages. This window displays the elements in a tree view, so that you can view the logical structure of the form or page and find controls that are deeply embedded or hidden.

See also

- [Class View and Object Browser icons](#)

Class View and Object Browser icons

10/18/2019 • 2 minutes to read • [Edit Online](#)

Class View and the **Object Browser** display icons that represent code entities, for example, namespaces, classes, functions, and variables. The following table illustrates and describes the icons.

ICON	DESCRIPTION	ICON	DESCRIPTION
Namespace	Method or Function	Method or Function	Method or Function
Class	Operator	Operator	Operator
Interface	Property	Property	Property
Structure	Field or Variable	Field or Variable	Field or Variable
Union	Event	Event	Event
Enum	Constant	Constant	Constant
TypeDef	Enum Item	Enum Item	Enum Item
Module	Map Item	Map Item	Map Item
Extension Method	External Declaration	External Declaration	External Declaration
Delegate	Error	Error	Error
Exception	Template	Template	Template
Map	Unknown	Unknown	Unknown
Type Forwarding			

Signal icons

The following signal icons apply to all the previous icons and indicate their accessibility.

ICON	DESCRIPTION
<No Signal Icon>	Public. Accessible from anywhere in this component and from any component that references it.
*	Protected. Accessible from the containing class or type, or those derived from the containing class or type.
#	Private. Accessible only in the containing class or type.

ICON	DESCRIPTION
⊗	Sealed.
♥	Friend/Internal. Accessible only from the project.
🔗	Shortcut. A shortcut to the object.

NOTE

If your project is included in a source control database, additional signal icons may be displayed to indicate source-control status, such as checked in or checked out.

See also

- [Viewing the structure of code](#)

Use the Task List

10/18/2019 • 3 minutes to read • [Edit Online](#)

Use **Task List** to track code comments that use tokens such as `TODO` and `HACK`, or custom tokens, and to manage shortcuts that take you directly to a predefined location in code. Click on the item in the list to go to its location in the source code.

NOTE

This topic applies to Visual Studio on Windows. For Visual Studio for Mac, see [Task comments \(Visual Studio for Mac\)](#).

The Task List window

When **Task List** is open, it appears at the bottom of the application window.

To open **Task List**, select **View > Task List**, or from the keyboard press **Ctrl+`T**.

Task List					
Description	Project	File	Line	Project Rank	
return _signInManager?? HttpContext.GetOwinContext().Get<ApplicationSignInManager>(); ConfigureAuth(app);		ManageController.cs	34	1	
		Startup.cs	18	1	
		Startup.cs	22	1	
TODO Add some logic here.	WebApplication1	ManageController.cs	21	1	
UNDONE Check-in #42	WebApplication1	ManageController.cs	26	1	
HACK temporary workaround	WebApplication1	ManageController.cs	33	1	
Note the authenticationType must match the one defined in CookieAuthenticationOptions.AuthenticationType	WebApplication1	IdentityModels.cs	14	1	

To change the sort order of the list, select the header of any column. To further refine your search results, press **Shift** and click a second column header. Alternatively, on the shortcut menu, choose **Sort by**, and then choose a header. To further refine your search results, press **Shift** and choose a second header.

To show or hide columns, on the shortcut menu, choose **Show Columns**. Select the columns that you want to show or hide.

To change the order of the columns, drag any column header to the location that you want.

User tasks

The user task feature was removed in Visual Studio 2015. When you open a solution that has user task data from Visual Studio 2013 and earlier, the user task data in your `.suo` file is not affected, but the user tasks are not displayed in the task list.

If you wish to continue to access and update your user task data, open the project in Visual Studio 2013 and copy the content of any user tasks into your preferred project management tool (such as Team Foundation Server).

Tokens and comments

A comment in your code preceded by a comment marker and a predefined token also appears in **Task List**. For example, the following C# comment has three distinct parts:

- The comment marker (`//`)
- The token, for example (`TODO`)
- The comment (the rest of the text)

```
// TODO: Load state from previously suspended application
```

Because `TODO` is a predefined token, this comment appears as a `TODO` task in the list.

NOTE

Default tokens are only available for the C/C++, C#, and VB languages. For other languages, see the [Custom tokens](#) section.

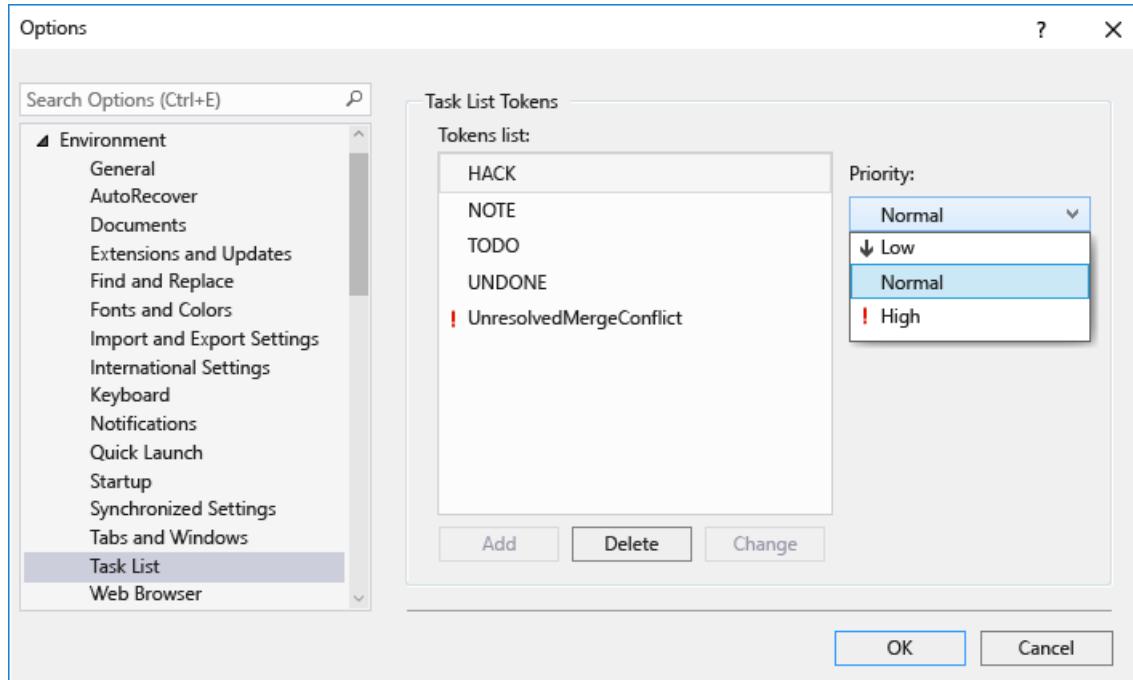
Custom tokens

By default, Visual Studio includes the following tokens: `HACK`, `TODO`, `UNDONE`, and `UnresolvedMergeConflict`. They are not case-sensitive. You can also create your own custom tokens.

To create a custom token:

1. On the **Tools** menu, choose **Options**.
2. Open the **Environment** folder and then choose **Task List**.

The [Task List options page](#) is displayed.



3. In the **Name** text box, enter your token name, for example **BUG**.
4. In the **Priority** drop-down list, choose a default priority for the new token.
5. Choose **Add**.

TIP

The **Add** button becomes enabled after you enter a name. You must enter a name before clicking **Add**.

C++ TODO comments

By default, C++ TODO comments are displayed in **Task List**.

To turn off C++ TODO comments, on the **Tools** menu, choose **Options > Text Editor > C/C++ > View > Enumerate Comment Tasks**, and set the value to **false**.

Shortcuts

A *shortcut* is a bookmark in the code that is tracked in **Task List**. It has a different icon than a regular bookmark. Double-click the shortcut in **Task List** to go to the corresponding location in the code.



Create a shortcut

To create a shortcut, insert the pointer into the code where you want to place a shortcut. Choose **Edit > Bookmarks > Add Task List Shortcut** or press **Ctrl+K, Ctrl+H**.

To navigate through the shortcuts in the code, choose a shortcut in the list, and then choose **Next Task** or **Previous Task** from the shortcut menu.

See also

- [Task List, Environment, Options dialog box](#)
- [Task comments \(Visual Studio for Mac\)](#)

Design and view classes and types with Class Designer

10/18/2019 • 2 minutes to read • [Edit Online](#)

Design, visualize, and refactor classes and other types in your code with **Class Designer** in Visual Studio. Use class diagrams to create and edit classes in your C#, Visual Basic, or C++ project. You can also use class diagrams to understand your project structure better or reorganize your code.

What you can do with class diagrams

- **Design:** Edit your project's code by editing the class diagram. Add new elements and delete unwanted ones. Your changes are reflected in code.
- **Visualize:** Understand your project's structure by viewing the classes in your project on a diagram. Customize your diagram so that you can focus on the project details that you care about the most. Save your diagram to use later for demonstration or documentation.
- **Refactor:** Override methods, rename identifiers, refactor parameters, and implement interfaces and abstract classes.

View types and relationships

Class diagrams show the details of types, for example, their constituent members, and the relationships between them. The visualization of these entities is a dynamic view into the code. This means that you can edit types in the designer and then see your edits reflected in the source code of the entity. Similarly, the class diagram is kept in sync with changes you make to code files.

NOTE

If your project contains a class diagram and your project references a type that's located in another project, the class diagram does not show the referenced type until you build the project for that type. Likewise, the diagram does not display changes to the code of the external entity until you rebuild the project for that entity.

Class diagram workflow

Class diagrams can help you understand the class structure of projects. These projects might have been created by other developers, or you just need a refresher on a project you created yourself. You can use class diagrams to customize, share, and present project information with others.

The first step in presenting project information is to create a class diagram that displays what you want to show. For more information, see [Add a class diagram](#). You can create multiple class diagrams for a project that can be used to display a distinct view of the project, a chosen subset of the project's types, or a chosen subset of the members of types.

In addition to defining what each class diagram shows, you can also change the way that information is presented; for more information, see [How to: Customize class diagrams](#).

After you have fine-tuned one or more class diagrams, you can copy them into Microsoft Office documents and print them, or export them as image files. For more information, see [How to: Copy class diagram elements to a Microsoft Office document](#), [How to: Print class diagrams](#) and [How to: Export class diagrams as images](#).

NOTE

Class Designer does not track the location of your source files, so changing your project structure or moving source files in the project can cause Class Designer to lose track of the type, especially the source type of a typedef, base classes, or association types. You might get an error, like **Class Designer is unable to display this type**. If you do, drag the modified or relocated source code to the class diagram again to redisplay it.

See also

- [Features of the code editor](#)
- [Map dependencies across your solutions](#)

How to: Add class diagrams to projects

10/31/2019 • 2 minutes to read • [Edit Online](#)

To design, edit, and refactor classes and other types, add a class diagram to your C#, Visual Basic, or C++ project. To visualize different parts of the code in a project, add multiple class diagrams to the project.

You can't create class diagrams from projects that share code across multiple apps. To create UML class diagrams, see [Create UML modeling projects and diagrams](#).

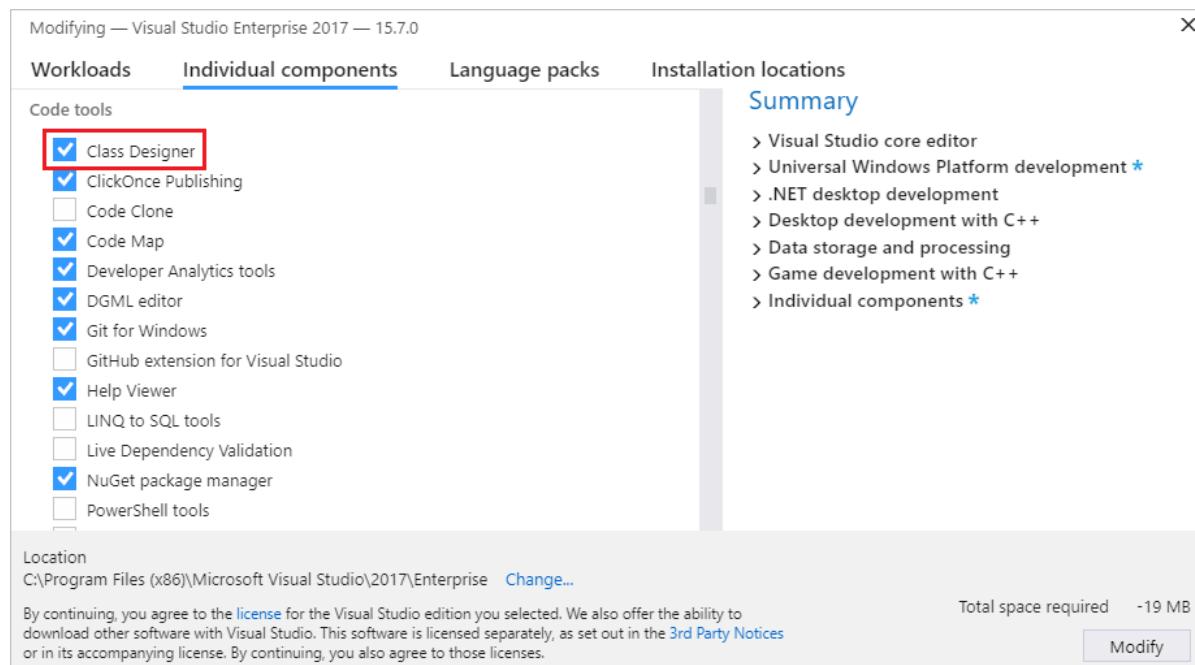
Install the Class Designer component

If you haven't installed the **Class Designer** component, follow these steps to install it.

1. Open **Visual Studio Installer** from the Windows Start menu, or by selecting **Tools > Get Tools and Features** from the menu bar in Visual Studio.

Visual Studio Installer opens.

2. Select the **Individual components** tab, and then scroll down to the **Code tools** category.
3. Select **Class Designer** and then select **Modify**.



The **Class Designer** component starts installing.

Add a blank class diagram to a project

1. In **Solution Explorer**, right-click the project node and then choose **Add > New Item**. Or, press **Ctrl+Shift+A**.

The **Add New Item** dialog opens.

2. Expand **Common Items > General**, and then select **Class Diagram** from the template list. For Visual C++ projects, look in the **Utility** category to find the **Class Diagram** template.

NOTE

If you don't see the **Class Diagram** template, [follow the steps](#) to install the **Class Designer** component for Visual Studio.

The class diagram opens in Class Designer and appears as a file that has a *.cd* extension in **Solution Explorer**. You can drag shapes and lines to the diagram from **Toolbox**.

To add multiple class diagrams, repeat the steps in this procedure.

Add a class diagram based on existing types

In **Solution Explorer**, open a class file's context menu (right-click) and then choose **View Class Diagram**.

-or-

In **Class View**, open the namespace or type context menu and then choose **View Class Diagram**.

TIP

If **Class View** is not open, open **Class View** from the **View** menu.

To display the contents of a complete project in a class diagram

In **Solution Explorer** or Class View, right-click the project and choose **View**, then choose **View Class Diagram**.

An auto-populated class diagram is created.

NOTE

Class Designer is not available in .NET Core projects.

See also

- [How to: Create types using the Class Designer](#)
- [How to: View existing types](#)
- [Design and view classes and types](#)

How to: Customize class diagrams

10/18/2019 • 3 minutes to read • [Edit Online](#)

You can change the way that class diagrams display information. You can customize the whole diagram or the individual types on the design surface.

For example, you can adjust the zoom level of an entire class diagram, change how individual type members are grouped and sorted, hide or show relationships, and move individual or sets of types anywhere on the diagram.

NOTE

Customizing the way that shapes appear on the diagram doesn't change the underlying code for the types represented on the diagram.

The sections that contain type members, such as the **Properties** section in a class, are called compartments. You can hide or show individual compartments and type members.

Zoom in and out of the class diagram

1. Open and select a class diagram file in **Class Designer**.
2. On the **Class Designer** toolbar, click the **Zoom In** or **Zoom Out** button to change the zoom level of the designer surface.

or

Specify a particular zoom value. You can use the **Zoom** drop down list or type a valid zoom level (valid range is between 10% and 400%).

NOTE

Changing the zoom level does not affect the scale of your class diagram printout.

Customize grouping and sorting of type members

1. Open and select a class diagram file in **Class Designer**.
2. Right-click an empty area on the design surface and point to **Group Members**.
3. Select one of the available options:
 - **Group by Kind** separates individual type members into a grouped list of Properties, Methods, Events, and Fields. The individual groups depend on the entities definition: for example, a class will not display any events group if there are no events yet defined for that class.
 - **Group by Access** separates individual type members into a grouped list based on the member's access modifiers. For example, Public and Private.
 - **Sort Alphabetically** displays the items that make up an entity as a single alphabetized list. The list is sorted in ascending order.

Hide compartments on a type

1. Open and select a class diagram file in the **Class Designer**.
2. Right click the member category in the type you want to customize (for example, select the **Methods** node in a class).
3. Click **Hide Compartment**.

The selected compartment disappears from the type container.

Hide individual members on a type

1. Open and select a class diagram file in **Class Designer**.
2. Right-click the member in the type you want to hide.
3. Click **Hide**.

The selected member disappears from the type container.

Show hidden compartments and members on a type

1. Open and select a class diagram file in **Class Designer**.
2. Right-click the name of the type with the hidden compartment.
3. Click **Show All Members**.

All hidden compartments and members appear in the type container.

Hide relationships

1. Open and select a class diagram file in **Class Designer**.
2. Right-click the association or inheritance line that you want to hide.
3. Click **Hide** for association lines, and click **Hide Inheritance Line** for inheritance lines.
4. Click **Show All Members**.

All hidden compartments and members appear in the type container.

Show hidden relationships

1. Open and select a class diagram file in **Class Designer**.
2. Right-click the type with the hidden association or inheritance.

Click **Show All Members** for association lines, and click **Show Base Class** or **Show Derived Classes** for inheritance lines.

Remove a shape from a class diagram

You can remove a type shape from the class diagram without affecting the type's underlying code. Removing type shapes from a class diagram affects only that diagram: the underlying code that defines the type and other diagrams that display the type are not affected.

1. On the class diagram, select the type shape you want to remove from the diagram.
2. On the **Edit** menu, choose **Remove from Diagram**.

The type shape and any lines of association or inheritance connected to the shape no longer appear on the

diagram.

Delete a type shape and its underlying code

1. Right-click the shape on the design surface.
2. Select **Delete Code** from the context menu.

The shape is removed from the diagram and its underlying code is deleted from the project.

See also

- [How to: Change Between Member Notation and Association Notation](#)
- [How to: View Existing Types](#)
- [Viewing Types and Relationships](#)

How to: Copy class diagram elements to a Microsoft Office document

10/18/2019 • 2 minutes to read • [Edit Online](#)

You can copy shapes from a .NET class diagram (.cd file) to other documents. You'll either get a copy of the shape or its underlying code, based on the kind of document where you paste it. To copy shapes from UML class diagrams in a modeling project, see [Export diagrams as images](#).

Copy a single element

Right-click the shape and choose **Copy Image**.

Copy several elements

1. Select the shapes on the diagram that you want to copy.
2. Right-click your selection and choose **Copy Image**.

Copy all the elements in a class diagram

1. Right-click the diagram surface and choose **Select All**, or press **Ctrl + A**.
2. On the **Edit** menu, select **Copy Image**.

You can also choose **Copy** instead of **Copy Image**. **Copy** copies the image as a regular bitmap. **Copy Image** copies the image as a vector-based image, which is better for most Office applications.

See also

- [How to: Print class diagrams](#)
- [How to: Export class diagrams as images](#)

How to: Export class diagrams as images

10/18/2019 • 2 minutes to read • [Edit Online](#)

To export a class diagram that you created from code in a project, save the diagram as an image. If you want to export UML class diagrams instead, see [Export diagrams as images](#).

Export a diagram

1. Open your class diagram (.cd) file.
2. From the **Class Diagram** menu or the diagram surface shortcut menu, choose **Export Diagram as Image**.
3. Select a diagram.
4. Select the format that you want.
5. Choose **Export** to finish exporting.

To automatically update exported images that are linked from other documents, export the diagram again in Visual Studio.

See also

- [How to: Print Class Diagrams](#)
- [Working with Class Diagrams](#)

How to: Print class diagrams

10/18/2019 • 2 minutes to read • [Edit Online](#)

You can print a class diagram using the print feature of Visual Studio.

To print a class diagram

1. Open the class diagram.
2. Click **Print** on the **File** menu.

The entire class diagram prints. You may need to adjust the settings in the **Page Setup** dialog box in order to print at an appropriate size.

See also

- [How to: Copy Class Diagram Elements to a Microsoft Office Document](#)
- [How to: Export Class Diagrams As Images](#)

How to: Add comments to class diagrams

10/18/2019 • 2 minutes to read • [Edit Online](#)

You can use comment shapes to annotate class diagrams. A comment shape has one property, **Text**, into which you can type text. Comment shapes exist only on the diagram surface and not in code.

A comment resides on the class diagram view in **Class Designer**. If you open a second class diagram onto the same project, comments you created in the first view are not visible. If you delete a diagram, all the comments it contained are also deleted.

You can resize a comment shape but you cannot change other aspects of its appearance, such as its background color, font, or font size.

To add a comment

1. Drag a comment from the **Class Designer Toolbox** onto the class diagram.
2. Click in the new comment shape on the diagram and type the text you want.

See also

- [Work with Class Diagrams](#)
- [How to: Customize Class Diagrams](#)

How to: Create types by using Class Designer

10/18/2019 • 2 minutes to read • [Edit Online](#)

To design new types for C# and Visual Basic projects, create them on a class diagram. To see existing types, see [How to: View Existing Types](#).

Create a new type

1. In the **Toolbox**, under **Class Designer**, drag one of these onto a class diagram:

- **Class or Abstract Class**
- **Enum**
- **Interface**
- **Structure (VB) or Struct (C#)**
- **Delegate**
- **Module (VB only)**

2. Name the type. Then select its access level.

3. Select the file where you want to add the initial code for the type:

- To create a new file and add it to the current project, select **Create new file** and name the file.
- To add code to an existing file, select **Add to existing file**.

If your solution has a project that shares code across multiple apps, you can add a new type to a class diagram in the app project, but only if the corresponding class file is in the same app project or is in the shared project.

4. Now add other items to define the type:

For	Add
Classes, abstract classes, structures or structs	Methods, properties, fields, events, constructors (method), destructors (method), and constants that define the type
Enums	Field values that make up the enumeration
Interfaces	Methods, properties, and events that make up the interface
Delegate	Parameters that define the delegate
Module	Methods, properties, fields, events, constructors (method), and constants that define the module

See [Creating Members](#).

Apply a custom attribute to a type

1. Click the type's shape on a class diagram.
2. In **Properties**, next to the **Custom Attributes** property for the type, click the ellipsis (...) button.
3. Add one or more custom attributes with one per line. Don't enclose them in brackets.

The custom attributes are applied to the type.

Apply a custom attribute to a type member

1. Click the member's name in its type's shape on a class diagram, or its row in the Class Details window.
2. In **Properties**, find the member's **Custom Attributes** property.
3. Add one or more custom attributes with one per line. Don't enclose them in brackets.

The custom attributes are applied to the type.

See also

- [How to: Create Inheritance Between Types](#)
- [How to: Create Associations Between Types](#)
- [Creating and Configuring Type Members](#)
- [Designing Classes and Types](#)

How to: Create inheritance between types in Class Designer

10/18/2019 • 2 minutes to read • [Edit Online](#)

To create an inheritance relationship between two types on a class diagram using **Class Designer**, connect the base type with its derived type or types. You can have an inheritance relationship between two classes, between a class and an interface, or between two interfaces.

To create an inheritance between types

1. From your project in **Solution Explorer**, open a class diagram (.cd) file.

If you don't have a class diagram, create it. See [How to: Add Class Diagrams to Projects](#).

2. In the **Toolbox**, under **Class Designer**, click **Inheritance**.

3. On the class diagram, draw an inheritance line between the types that you want, starting from:

- A derived class to the base class
- An implementing class to the implemented interface
- An extending interface to the extended interface

4. Optionally, when you have a derived type from a generic type, click the inheritance line. In the **Properties** window, set the **Type Arguments** property to match the type that you want for the generic type.

NOTE

If a parent abstract class contains at least one abstract member, then all abstract members are implemented as non-abstract inheriting classes.

Although you can visualize existing generic types, you can't create new generic types. You also can't change the type parameters for existing generic types.

See also

- [Inheritance](#)
- [Inheritance Basics](#)
- [How to: View Inheritance Between Types](#)
- [Visual C++ Classes in Class Designer](#)

How to: Create associations between types in Class Designer

10/18/2019 • 2 minutes to read • [Edit Online](#)

Association lines in **Class Designer** show how classes in a diagram are related. An Association line represents a class that is the type of a property or field of another class in your project. Association lines are generally used to illustrate the most important relationships between classes in your project.

While you could display all fields and properties as associations, it makes more sense to show only important members as associations, depending on what you intend to emphasize in the diagram. (You can show less important members as regular members or hide them altogether.)

NOTE

Class Designer supports only unidirectional associations.

To define an association line in the Class Diagram

1. In the Toolbox, under **Class Designer**, select **Association**.
2. Draw a line between the two shapes you want to link with an association.

A new property is created in the first class. This property displays as an association line (not as a property within a compartment in the shape) with a default name. Its type is the shape to which the association line points.

To change the name of an association

On the diagram surface, click the label of the association line and edit it.

Alternatively, follow these steps:

1. Select the shape that contains the property that is shown as an association.

The shape obtains focus and its members display in the **Class Details** and **Properties** windows.

2. In either the **Class Details** or **Properties** window, edit the name field for that property and press **Enter**.

The name is updated in the **Class Details** window, on the association line, in the **Properties** window, and in code.

See also

- [How to: Change between member notation and association notation](#)

How to: Visualize a collection association in Class Designer

10/18/2019 • 2 minutes to read • [Edit Online](#)

Properties and fields that are collections of other types can be displayed on the class diagram as a collection association. Unlike a regular association, which displays a field or property as a line linking the owning class to the field's type, a collection association is displayed as a line linking the owning class to the collected type.

To create a collection association

1. In code, create a property or field whose type is itself a strongly-typed collection.
2. In the class diagram, expand the class so that properties and fields are shown.
3. In the class, right-click the field or property and choose **Show as Collection Association**.

The property or field is shown as an association line linking to the collected type.

See also

- [How to: Create Associations Between Types](#)
- [Designing Classes and Types](#)

Create and configure type members in Class Designer

10/18/2019 • 13 minutes to read • [Edit Online](#)

You can add these members to types on a class diagram and configure those members in the **Class Details** window:

TYPE	MEMBERS IT CAN CONTAIN
Class	method, property (for C# and Visual Basic), field, event (for C# and Visual Basic), constructor (method), destructor (method), constant
Enum	member
Interface	method, property, event (for C# and Visual Basic)
Abstract Class	method, property (for C# and Visual Basic), field, event (for C# and Visual Basic), constructor (method), destructor (method), constant
Structure (Struct in C#)	method, property (for C# and Visual Basic) field, event (for C# and Visual Basic), constructor (method), constant
Delegate	parameter
Module (VB Only)	method, property, field, event, constructor, constant

NOTE

Make property declaration more concise when a property's get and set accessors don't need additional logic by using auto-implemented properties (C# only). To show the full signature, from the **Class Diagram** menu choose **Change Members Format > Display Full Signature**. For more information about auto-implemented properties, see [Auto-Implemented Properties](#).

Common tasks

TASK	SUPPORTING CONTENT
Get started: Before you create and configure type members, you must open the Class Details window.	<ul style="list-style-type: none">- Open the Class Details window- Class Details usage notes- Display of read-only information- Keyboard and mouse shortcuts in the Class Diagram and Class Details window
Create and modify type members: You can create new members, modify members, and add parameters to a method by using the Class Details window.	<ul style="list-style-type: none">- Create members- Modify type members- Add parameters to methods

Open the Class Details window

By default, the **Class Details** window appears automatically when you open a new class diagram. See [How to: Add class diagrams to projects](#)). You can also open the **Class Details** window in the following ways:

- Right-click on any class in the diagram to display a context menu, and then select **Class Details**.
- Select **View > Other Windows > Class Details** from the menu bar.

Create members

You can create a member using any of the following tools:

- **Class Designer**
- **Class Details** window toolbar
- **Class Details** window

NOTE

You can also create constructors and destructors using the procedures in this section. Please bear in mind that constructors and destructors are special kinds of methods, and as such, they appear in the **Methods** compartment in class diagram shapes and in the **Methods** section of the **Class Details** window grid.

NOTE

The only entity you can add to a delegate is parameter. Note that the procedure entitled 'To Create a member using the **Class Details** window toolbar' is not valid for this action.

Create a member using Class Designer

1. Right-click the type to which you want to add a member, point to **Add**, and then choose the type of member you want to add.

A new member signature is created and added to the type. It is given a default name that you can change in **Class Designer**, the **Class Details** window, or in the **Properties** window.

2. Optionally, specify other details about the member, such as its type.

Create a member using the Class Details window toolbar

1. On the diagram surface, select the type to which you want to add a member.

The type obtains focus and its contents are displayed in the **Class Details** window.

2. In the **Class Details** window toolbar, click the top icon and select **New <member>** from the drop-list.

The cursor moves to the **Name** field in a row for the kind of member you want to add. For example, if you clicked **New Property**, the cursor moves to a new row in the **Properties** section of the **Class Details** window.

3. Type the name of the member you want to create and press Enter (or otherwise move focus, such as by pressing Tab).

A new member signature is created and added to the type. The member now exists in code and is displayed in **Class Designer**, the **Class Details** window, and the **Properties** window.

4. Optionally, specify other details about the member, such as its type.

Create a member using the Class Details window

1. On the diagram surface, select the type to which you want to add a member.

The type obtains focus and its contents are displayed in the **Class Details** window.

2. In the **Class Details** window, in the section that contains the kind of member you want to add, click <**add member**>. For example, if you want to add a field, click <**add field**>.

3. Type the name of the member you want to create and press Enter.

A new member signature is created and added to the type. The member now exists in code and is displayed in the **Class Designer**, the **Class Details** window, and the Properties window.

4. Optionally, specify other details about the member, such as its type.

NOTE

You can also use keyboard shortcuts to create members. For more information, see [Keyboard and Mouse Shortcuts in the Class Diagram and Class Details window](#).

Modify type members

Class Designer enables you to modify the members of types that are displayed on the diagram. You can modify the members of any type displayed on a class diagram that are not read-only. You modify type members by using in-place editing on the design surface, Properties window, and the **Class Details** window.

All the members displayed in the **Class Details** window represent the members of the types on the class diagram. There are four kinds of members: methods, properties, fields, and events.

All member rows appear under headings that group the members by kind. For example, all properties appear under the heading **Properties**, which, as a node in the grid, can be collapsed or expanded.

Each member row displays the following elements:

- **Member Icon**

Each kind of member is represented by its own icon. Point the mouse at the member icon to display the member's signature. Click the member icon or the whitespace to the left of the member icon to select the row.

- **Member Name**

The **Name** column in a member row displays the name of the member. This name is also displayed in the **Name** property in the Properties window. Use this cell to change the name of any member that has read-write permissions.

If the **Name** column is too narrow to show the whole name, pointing the mouse on the member name displays the entire name.

- **Member Type**

The **MemberType** cell uses IntelliSense, which lets you select from a list of all the types available in the current project or referenced projects.

- **Member Modifier**

Change the visibility modifier of a member to either `Public` (`public`), `Private` (`private`), `Friend` (`internal`), `Protected` (`protected`), `Protected Friend` (`protected internal`), or `Default`.

- <add member>

The last row in the **Class Details** window contains the text **<add member>** in the **Name** cell. If you click this cell, you can create a new member. For more information, see [Create members](#).

- Member properties in the Properties window

The **Class Details** window displays a subset of the member properties that are displayed in the Properties window. Changing a property in one location will update the value of the property globally. This includes the display of its value in the other location.

- Summary

The **Summary** cell exposes a summary of information about the member. Click the ellipsis in the **Summary** cell to view or edit information about the **Summary**, **Return Type**, and **Remarks** for the member.

- Hide

When the **Hide** check box is selected, the member is not displayed in the type.

To modify a type member

1. Using Class Designer, select a type.
2. If the **Class Details** window is not displayed, click the **Class Details** window button on the Class Designer toolbar.
3. Edit the values in the fields of the **Class Details** window grid. After each edit, press ENTER, or otherwise move focus away from the edited field, for example, by pressing TAB. Your edits reflect immediately in code.

NOTE

If you want to modify only the name of a member, you can do so by using in-place editing.

Add parameters to methods

Add parameters to methods using the **Class Details** window. Parameters can be configured to be required or optional. Providing a value for the **Optional Default** property of a parameter instructs the designer to generate code as an optional parameter.

Parameter rows contain the following items:

- Name

The **Name** column in a parameter row displays the name of the parameter. This name is also displayed in the **Name** property in the Properties window. You can use this cell to change the name of any parameter with read-write permissions.

Pointing at the parameter name displays the name of the parameter if the **Name** column is too narrow to show the entire name.

- Type

The **Parameter Type** cell uses IntelliSense, which lets you choose from a list of all the types available in the current project or referenced projects.

- Modifier

The **Modifier** cell in a parameter row accepts and displays the new modifier of the parameter. To enter a

new parameter modifier, use the drop-down list box to select from **None**, **ref**, **out**, or **params** in C#, and **ByVal**, **ByRef**, or **ParamArray** in VB.

- **Summary**

The **Summary** cell in a parameter row allows entering of code comments that appear in IntelliSense when entering the parameter into the code editor.

- **<add parameter>**

The last parameter row of a member contains the text **<add parameter>** in the **Name** cell. Clicking this cell lets you create a new parameter. For more information, see [To add a parameter to a method](#).

The **Properties** window displays the same parameter properties displayed in the **Class Details** window: **Name**, **Type**, **Modifier**, **Summary**, as well as the **Optional Default** property. Changing a property in one location updates the value of the property globally, including the display of its value in the other location.

NOTE

To add a parameter to a delegate, see [Create members](#).

NOTE

Although a destructor is a method, it cannot have parameters.

To add a parameter to a method

1. On the diagram surface, click the type containing the method to which you want to add a parameter.

The type obtains focus and its contents display in the **Class Details** window.

2. In the **Class Details** window, expand the row of the method to which you want to add a parameter.

An indented parameter row appears, containing only a pair of parentheses and the words **<add parameter>**.

3. Click **<add parameter>**, type the name of the new parameter, and press **Enter**.

The new parameter is added to the method and the method's code. It displays in the **Class Details** window and the Properties window.

4. Optionally, specify other details about the parameter, such as its type.

To add an optional parameter to a method

1. On the diagram surface, click the type containing the method to which you want to add an optional parameter.

The type obtains focus and its contents display in the **Class Details** window.

2. In the **Class Details** window, expand the row of the method to which you want to add an optional parameter.

An indented parameter row appears, containing only a pair of parentheses and the words **<add parameter>**.

3. Click **<add parameter>**, type the name of the new parameter, and press **Enter**.

The new parameter is added to the method and the method's code. It displays in the **Class Details** window and the Properties window.

4. In the Properties window, type a value for the **Optional Default** property. Setting a parameter's Optional Default property makes that parameter optional.

NOTE

Optional parameters must be the last parameters in the parameter list.

Class details usage notes

Please note the following tips for using the **Class Details** window.

Editable and non-editable cells

All cells in the **Class Details** window are editable with a few exceptions:

- The entire type is read-only, when, for example, it resides in a referenced assembly. When you select the shape in the Class Designer, the **Class Details** window displays its details in a read-only state.
- For indexers, the name is read-only and the rest (type, modifier, summary) are editable.
- All generics have read-only parameters in the **Class Details** window. To change a generic parameter, edit its source code.
- The name of the type parameter that is defined on a generic type is read-only.
- When a type's code is broken (unparsable), **Class Details** window displays the type's contents as read-only.

The Class Details window and source code

- You can view source code by right-clicking a shape in the **Class Details** window (or the Class Designer) and then clicking View Code. The source code file opens and scrolls to the selected element.
- Changing source code is immediately reflected in the display of signature information in the Class Designer and the **Class Details** window. If the **Class Details** window is closed at the time, the new information is visible the next time you open it.
- When a type's code is broken (unparsable), **Class Details** window displays the type's contents as read only.

Clipboard functionality in the Class Details window

You can copy or cut fields or rows from the **Class Details** window and paste them into another type. You can cut a row only if it is not read-only. When you paste the row, **Class Details** window assigns a new name (derived from the name of the copied row) to avoid a conflict.

Display of read-only information

Class Designer and the **Class Details** window can display the types (and members of types) for the following:

- a project that contains a class diagram
- a project referenced from a project that contains a class diagram
- an assembly referenced from a project that contains a class diagram

In the latter two cases, the referenced entity (a type or member) is read-only in the class diagram that represents it.

An entire project or portions of it, such as individual files, may be read-only. The most common cases in which a project or one of its files is read-only are when it is under source-code control (and not checked out), it exists in an external assembly, or when the operating system considers the files to be read-only.

Source-Code Control

Because a class diagram is saved as a file in a project, you need to check out the project in order to save any changes you make in Class Designer or the **Class Details** window.

Read-Only Projects

The project may be read-only for a reason other than source-code control. Closing the project displays a dialog box asking whether to overwrite the project file, discard changes (don't save) or cancel the close operation. If you choose to overwrite, project files are overwritten and made read-write. The new class diagram file is added.

Read-Only Types

If you try to save a project containing a type whose source-code file is read-only, the **Save of Read-Only File** dialog box appears, which gives you choices to save the file under a new name or new location, or to overwrite the read-only file. If you overwrite the file, the new copy is no longer read-only.

If a code file contains a syntax error, shapes displaying code in that file will be temporarily read-only until the syntax error is fixed. Shapes in this state display red text and a red icon which displays a tooltip reading "The source code file contains a parse error".

A referenced type (such as a .NET type), which exists under another project node or under a referenced-assembly node, is indicated on the Class Designer design surface as read-only. A local type, which exists in the project you have open, is read-write, and its shape on the Class Designer design surface is indicated as such.

Indexers are read-write in code and the **Class Details** window, but the indexer name is read-only.

You cannot edit partial methods by using the Class Designer or the **Class Details** window; you must use the Code Editor to edit them.

You cannot edit native C++ code by using the Class Designer or the **Class Details** window; you must use the Code Editor to edit native C++ code.

See also

- [Viewing types and relationships](#)
- [Refactoring classes and types](#)

How to: View existing types in Class Designer

10/18/2019 • 2 minutes to read • [Edit Online](#)

To see an existing type and its members, add its shape to a class diagram.

You can see local and referenced types. A local type exists in the currently open project and is read/write. A referenced type exists in another project or in a referenced assembly and is read-only.

To design new types on class diagrams, see [How to: Create types by using Class Designer](#).

To see types in a project on a class diagram

1. From a project in **Solution Explorer**, open an existing class diagram (.cd) file. Or if no class diagram exists, add a new class diagram to the project. See [How to: Add Class Diagrams to Projects](#).
2. From the project in **Solution Explorer**, drag a source code file to the class diagram.

NOTE

If your solution has a project that shares code across multiple apps, you can drag files or code to a class diagram only from these sources:

- The app project that contains the diagram
- A shared project that was imported by the app project
- A referenced project
- An assembly

Shapes representing the types defined in the source code file appear on the diagram at the position where you dragged the file.

You can also view types in the project by dragging one or more types from the project node in **Class View** to the class diagram.

TIP

If **Class View** is not open, open **Class View** from the **View** menu.

To display types at default locations on the diagram, select one or more types in **Class View**, right-click the selected types, and choose **View Class Diagram**.

NOTE

If a closed class diagram containing the type already exists in the project, the class diagram opens to display the type shape. However, if no class diagram containing the type exists in the project, **Class Designer** creates a new class diagram in the project and opens it to display the type.

When you first display a type on the diagram, its shape appears collapsed by default. You can expand the shape to view its contents.

To display the contents of a project in a class diagram

In **Solution Explorer** or **Class View**, right-click the project and choose **View**, then choose **View Class Diagram**.

An auto-populated Class Diagram is created.

See also

- [How to: View Inheritance Between Types](#)
- [How to: Customize Class Diagrams](#)
- [Viewing Types and Relationships](#)

How to: View inheritance between types in Class Designer

10/18/2019 • 2 minutes to read • [Edit Online](#)

You can find the inheritance relationship, if it exists, between a base type and its derived types on a class diagram in **Class Designer**. To create an inheritance relationship, if none exist, between two types, see [How to: Create inheritance between types](#).

To find the base type

1. On the class diagram, click the type for which you want to see the base class or interface.
2. On the **Class Diagram** menu, choose **Show Base Class** or **Show Base Interfaces**.

The type's base class or interface appears selected on the diagram. Any hidden inheritance lines now appear between the two shapes.

You can also right-click the type whose base type you want to display, and choose **Show Base Class** or **Show Base Interfaces**.

To find the derived types

1. On the class diagram, click the type for which you want to see the derived classes or interfaces.
2. On the **Class Diagram** menu, choose **Show Derived Classes** or **Show Derived Interfaces**.

The type's derived classes or interfaces appear on the diagram. Any hidden inheritance lines now appear between the shapes.

You can also right-click the type for which you want to see its derived types, and choose **Show Derived Classes** or **Show Derived Interfaces**.

See also

- [How to: Create Associations Between Types](#)
- [Viewing Types and Relationships](#)

How to: Change between member notation and association notation in Class Designer

10/18/2019 • 2 minutes to read • [Edit Online](#)

In **Class Designer**, you can change the way the class diagram represents an association relationship between two types from member notation to association notation and vice versa. Members displayed as association lines often provide a useful visualization of how types are related.

NOTE

Association relationships can be represented as a member property or field. To change member notation to association notation, one type must have a member of another type. To change association notation to member notation, the two types must be connected by an association line. For more information, see [How to: Create associations between types](#). If your project contains multiple class diagrams, changes that you make to the way a diagram displays association relationships affect only that diagram. To change the way another diagram displays association relationships, open or display that diagram and perform these steps.

To change member notation to association notation

1. From the project node in Solution Explorer, open the class diagram (.cd) file.
2. In the type shape on the class diagram, right-click the member property or field representing the association, and choose **Show as Association**.

TIP

If no properties or fields are visible in the type shape, the compartments in the shape might be collapsed. To expand the type shape, double-click the compartment name or right-click the type shape, and choose **Expand**.

The member disappears from the compartment in the type shape and an association line appears to connect the two types. The association line is labeled with the name of the property or field.

To change association notation to member notation

On the class diagram, right-click the association line, and choose **Show as Property** or **Show as Field** as appropriate. The association line disappears, and the property displays in the appropriate compartment within its type shape on the diagram.

See also

- [How to: Create Inheritance Between Types](#)
- [How to: View Inheritance Between Types](#)
- [Viewing Types and Relationships](#)
- [How to: Visualize a Collection Association](#)

Refactor classes and types in Class Designer

10/18/2019 • 2 minutes to read • [Edit Online](#)

When you refactor code, you make it easier to understand, maintain, and more efficient by changing its internal structure and how its objects are designed, not its external behavior. Use Class Designer and the Class Details window to reduce the work that you have to do and the chance of introducing bugs when you refactor C#, Visual Basic, or C++ code in your Visual Studio project.

NOTE

The files of a project might be read-only because the project is under source-code control and is not checked out, it is a referenced project, or its files are marked as read-only on disk. When you work in a project in one of these states, you will be presented with various ways to save your work depending on the project's state. This applies to refactoring code and also to code that you change in another way, such as directly editing it.

Common tasks

TASK	SUPPORTING CONTENT
Refactoring classes: You can use refactoring operations to split a class into partial classes or to implement an abstract base class.	- How to: Split a Class into Partial Classes
Working with interfaces: In Class Designer, you can implement an interface on the class diagram by connecting it to a class that provides code for the interface methods.	- How to: Implement an Interface
Refactoring types, type members, and parameters: By using Class Designer, you can rename types, override type members, or move them from one type to another. You can also create nullable types.	- Rename types and type members - Move type members from one type to another - How to: Create a Nullable Type

Rename types and type members

In Class Designer, you can rename a type or a member of a type on the class diagram or in the **Properties** window. In the **Class Details** window, you can change the name of a member but not a type. Renaming a type or type member propagates to all windows and code locations where the old name appeared.

Rename in the Class Designer

1. On the class diagram, select the type or member and select the name.

The name of the member becomes editable.

2. Type the new name for the type or type member

Rename in the Class Details window

1. To display the **Class Details** window, right-click the type or type member and select **Class Details**.

The **Class Details** window appears.

2. In the **Name** column, change the name of the type member

3. To move focus away from the cell, press the **Enter** key or click away from the cell.

NOTE

In the **Class Details** window, you can change the name of a member but not a type.

Rename in the Properties window

1. On the class diagram or the **Class Details** window, right-click the type or member and then select **Properties**.

The **Properties** window appears and displays properties for the type or type member.

2. In the **Name** property, change the name of the type or type member.

The new name propagates to all windows and code locations in the current project where the old name appeared.

Move type members from one type to another

Using **Class Designer**, you can move a type member from one type to another type. Both types must be visible in the current class diagram.

1. In a type that is visible on the design surface, right-click the member you want to move to another type, and then select **Cut**.
2. Right-click the destination type and select **Paste**.

The property is removed from the source type and appears in the destination type.

See also

- [Designing Classes and Types](#)

How to: Implement an interface in Class Designer

10/18/2019 • 2 minutes to read • [Edit Online](#)

In **Class Designer**, you can implement an interface on the class diagram by connecting it to a class that provides code for the interface methods. **Class Designer** generates an interface implementation and displays the relationship between the interface and the class as an inheritance relationship. You can implement an interface by drawing an inheritance line between the interface and the class or by dragging the interface from Class View.

TIP

You can create interfaces the same way you create other types. If the interface exists but does not appear on the class diagram, then first display it. For more information, see [How to: Create types by using Class Designer](#) and [How to: View existing types](#).

To implement an interface by drawing an inheritance line

1. On the class diagram, display the interface and the class that will implement the interface.
2. Draw an inheritance line from the class and the interface.

A lollipop appears attached to the class and a label with the interface name identifies the inheritance relationship. Visual Studio generates stubs for all interface members.

For more information, see [How to: Create inheritance between types](#).

To implement an interface from the Class View window

1. On the class diagram, display the class that you want to implement the interface.
2. Open **Class View** and locate the interface.

TIP

If **Class View** is not open, open **Class View** from the **View** menu or press **Ctrl+Shift+C**.

3. Drag the interface node to the class shape on the diagram.

A lollipop appears attached to the class and a label with the interface name identifies the inheritance relationship. Visual Studio generates stubs for all interface members; at this point, the interface is implemented.

See also

- [How to: Create Types by using Class Designer](#)
- [How to: View Existing Types](#)
- [How to: Create Inheritance Between Types](#)
- [Refactoring Classes and Types](#)

How to: Split a class into partial classes in Class Designer

10/18/2019 • 2 minutes to read • [Edit Online](#)

You can use the `partial` keyword (`Partial` in Visual Basic) to divide the declaration of a class or structure among several declarations. You can use as many partial declarations as you want.

The declarations can be in one or in multiple source files. All the declarations must be in the same assembly and the same namespace.

Partial classes are useful in several situations. On a large project, for example, separating a class into multiple files enables more than one programmer to work on the project at same time. When you're working with code that Visual Studio generates, you can change the class without having to re-create the source file. (Examples of code that Visual Studio generates include Windows Forms and web service wrapper code.) You can thus create code that uses auto-generated classes without having to modify the file that Visual Studio creates.

There are two kinds of partial methods. In C#, they are called declaring and implementing; in Visual Basic, they are called declaration and implementation.

Class Designer supports partial classes and methods. The type shape in the class diagram refers to a single declaration location for the partial class. If the partial class is defined in multiple files, you can specify which declaration location **Class Designer** will use by setting the **New Member Location** property in the **Properties** window. That is, when you double-click a class shape, **Class Designer** goes to the source file that contains the class declaration identified by the **New Member Location** property. When you double-click a partial method in a class shape, **Class Designer** goes to the partial method declaration. Also, in the **Properties** window, the **File Name** property refers to the declaration location. For partial classes, **File Name** lists all of the files that contain declaration and implementation code for that class. However, for partial methods, **File Name** lists only the file that contains the partial method declaration.

The following examples split the definition of class `Employee` into two declarations, each of which defines a different procedure. The two partial definitions in the examples could be in one source file or in two different source files.

NOTE

Visual Basic uses partial-class definitions to separate Visual Studio-generated code from user-authored code. The code is separated into discrete source files. For example, the **Windows Form Designer** defines partial classes for controls such as `Form`. You should not modify the generated code in these controls.

For more information about partial types in Visual Basic, see [Partial](#).

Example

To split a class definition, use the `partial` keyword (`Partial` in Visual Basic), as shown in the following example:

```
// First part of class definition.  
public partial class Employee  
{  
    public void CalculateWorkHours()  
    {  
    }  
}  
  
// Second part of class definition.  
public partial class Employee  
{  
    public void CalculateTaxes()  
    {  
    }  
}
```

```
' First part of class definition.  
Partial Public Class Employee  
    Public Sub CalculateWorkHours()  
    End Sub  
End Class  
  
' Second part of class definition.  
Partial Public Class Employee  
    Public Sub CalculateTaxes()  
    End Sub  
End Class
```

See also

- [Partial classes and methods](#)
- [partial \(Type\) \(C# Reference\)](#)
- [partial \(Method\) \(C# Reference\)](#)
- [Partial \(Visual Basic\)](#)

How to: Create a nullable type in Class Designer

10/18/2019 • 3 minutes to read • [Edit Online](#)

Certain value types do not always have (or need) a defined value. This is common practice in databases, where some fields might not be assigned any value. For example, you might assign a null value to a database field to signify that it has not yet been assigned a value.

A *nullable type* is a value type that you extend so that it takes the typical range of values for that type and also a null value. For example, a nullable of `Int32`, also denoted as `Nullable<Int32>`, can be assigned any value from -2147483648 to 2147483647, or it can be assigned a null value. A `Nullable<bool>` can be assigned the values `True`, `False`, or `null` (no value at all).

Nullable types are instances of the `Nullable<T>` structure. Each instance of a nullable type has two public read-only properties, `HasValue` and `Value`:

- `HasValue` is of type `bool` and indicates whether the variable contains a defined value. `True` means that the variable contains a non-null value. You can test for a defined value by using a statement such as `if (x.HasValue)` or `if (y != null)`.
- `Value` is of the same type as the underlying type. If `HasValue` is `True`, `Value` contains a meaningful value. If `HasValue` is `False`, accessing `Value` will throw an invalid operation exception.

By default, when you declare a variable as a nullable type, it has no defined value (`HasValue` is `False`), other than the default value of its underlying value type.

Class Designer displays a nullable type just as it displays its underlying type.

For more information about nullable types in C#, see [Nullable Types](#). For more information about nullable types in Visual Basic, see [Nullable Value Types](#).

NOTE

Your computer might show different names or locations for some of the Visual Studio user interface elements in this article. You may be using a different edition of Visual Studio or different environment settings. For more information, see [Personalize the IDE](#).

To add a nullable type by using the Class Designer

1. In the class diagram, expand an existing class or create a new class.
2. To add a class to the project, on the **Class Diagram** menu, click **Add > Add Class**.
3. To expand the class shape, on the **Class Diagram** menu, click **Expand**.
4. Select the class shape. On the **Class Diagram** menu, click **Add > Field**. A new field that has the default name **Field** will appear in the class shape and also in the **Class Details** window.
5. In the **Name** column of the **Class Details** window (or in the class shape itself), change the name of the new field to a valid and meaningful name.
6. In the **Type** column of the **Class Details** window, declare the type as a nullable type by specifying the following:
 - For **System.Int32**, type `Nullable<Int32>`.
 - For **System.Boolean**, type `Nullable<bool>`.

- `int?` (Visual C#)
- `Nullable(Of Integer)` (Visual Basic)

To add a nullable type by using the Code Editor

1. Add a class to the project. Select the project node in **Solution Explorer**, and, on the **Project** menu, click **Add Class**.
2. In the .cs or .vb file for the new class, add one or more nullable types in the new class to the class declaration.

```
// Declare a nullable type in Visual C#:  
class Test  
{  
    int? building_number = 5;  
}
```

```
' Declare a nullable type in Visual Basic:  
Class Test  
    Dim buildingNumber As Nullable(Of Integer) = 5  
End Class
```

3. From Class View, drag the new class icon to the Class Designer design surface. A class shape appears in the class diagram.
4. Expand the details for the class shape and move the mouse pointer over the class members. The tooltip displays the declaration of each member.
5. Right-click the class shape and click **Class Details**. You can view or modify the new type's properties in the **Class Details** window.

See also

- [Nullable<T>](#)
- [Nullable Types](#)
- [Using Nullable Types](#)
- [How to: Identify a Nullable Type](#)
- [Nullable Value Types](#)

Work with C++ code in Class Designer

10/31/2019 • 5 minutes to read • [Edit Online](#)

Class Designer displays a visual design surface called a *class diagram* that provides a visual representation of the code elements in your project. You can use class diagrams to design and visualize classes and other types in a project.

Class Designer supports the following C++ code elements:

- Class (resembles a managed class shape, except that it can have multiple inheritance relationships)
- Anonymous class (displays Class View's generated name for the anonymous type)
- Template class
- Struct
- Enum
- Macro (displays the post-processed view of the macro)
- Typedef

NOTE

This is not the same as the UML class diagram, which you can create in a Modeling Project. For more information, see [UML Class Diagrams: Reference](#).

Troubleshoot type resolution and display issues

Location of source files

Class Designer does not keep track of the location of source files. Therefore, if you modify your project structure or move source files in your project, **Class Designer** can lose track of the type (especially the source type of a typedef, base classes, or association types). You might receive an error such as **Class Designer is unable to display this type**. If you do, drag the modified or relocated source code to the class diagram again to redisplay it.

Update and performance issues

For C++ projects, it might take 30 to 60 seconds for a change in the source file to appear in the class diagram. This delay might also cause **Class Designer** to throw the error **No types were found in the selection**. If you receive an error such as this, click **Cancel** in the error message and wait for the code element to appear in **Class View**.

After you do this, **Class Designer** should be able to display the type.

If a class diagram does not update with changes you have made in the code, you might need to close the diagram and open it again.

Type resolution issues

Class Designer might not be able to resolve types for the following reasons:

- The type is in a project or assembly that is not referenced from the project that contains the class diagram.
To correct this error, add a reference to the project or assembly that contains the type. For more information, see [Managing references in a project](#).
- The type is not in the correct scope, so **Class Designer** cannot locate it. Ensure that the code is not missing

a `using`, `imports`, or `#include` statement. Also make sure that you have not moved the type (or a related type) out of the namespace in which it was originally located.

- The type does not exist (or has been commented out). To correct this error, make sure that you have not commented out or deleted the type.
- The type is located in a library referenced by an `#import` directive. A possible workaround is to manually add the generated code (the `.tlh` file) to an `#include` directive into the header file.
- Ensure that **Class Designer** supports the type that you entered. See [Limitations for C++ Code Elements](#).

The error you are most likely to see for a type resolution issue is **Code could not be found for one or more shapes in class diagram '<element>'**. This error message does not necessarily indicate that your code is in error. It indicates only that class designer was unable to display your code. Try the following measures:

- Ensure that the type exists. Ensure that you have not unintentionally commented out or deleted the source code.
- Try to resolve the type. The type might be in a project or assembly that is not referenced from the project that contains the class diagram. To correct this error, add a reference to the project or assembly that contains the type. For more information, see [Managing references in a project](#).
- Ensure that the type is in the correct scope so that Class Designer can locate it. Make sure that the code is not missing a `using`, `imports`, or `#include` statement. Also make sure that you have not moved the type (or a related type) out of the namespace in which it was originally located.

Troubleshoot other error messages

You can find assistance with troubleshooting errors and warnings in the Microsoft Developer Network (MSDN) public forums. See the [Visual Studio Class Designer Forum](#).

Limitations for C++ code elements

- When a C++ project is loaded, **Class Designer** functions in a read-only manner. You can change the class diagram, but you cannot save changes from the class diagram back to the source code.
- **Class Designer** supports only native C++ semantics. For C++ projects that are compiled into managed code, **Class Designer** will only visualize code elements that are native types. Therefore, you can add a class diagram to a project, but **Class Designer** will not allow you to visualize elements in which the `IsManaged` property is set to `true` (that is, value types and reference types).
- For C++ projects, the **Class Designer** reads only the definition of the type. For example, assume that you define a type in a header (.h) file and define its members in an implementation (.cpp) file. If you invoke "View Class Diagram" on the implementation (.cpp) file, **Class Designer** displays nothing. As another example, if you invoke "View Class Diagram" on a .cpp file that uses an `#include` statement to include other files but does not contain any actual class definitions, **Class Designer** again displays nothing.
- IDL (.idl) files, which define COM interfaces and type libraries, do not display in diagrams unless they are compiled to native C++ code.
- **Class Designer** does not support global functions and variables.
- **Class Designer** does not support unions. This is a special type of class in which the memory allocated is only the amount necessary for the union's largest data member.
- **Class Designer** does not display basic data types such as `int` and `char`.
- **Class Designer** does not display types that are defined outside the current project if the project does not have correct references to those types.

- **Class Designer** can display nested types but not the relationships between a nested type and other types.
- **Class Designer** cannot display types that are void or that derive from a void type.

See also

- [Designing and Viewing Classes and Types](#)
- [Additional Information About Class Designer Errors](#)
- [C++ Classes in Class Designer](#)
- [C++ Structures in Class Designer](#)
- [C++ Enumerations in Class Designer](#)
- [C++ Typedefs in Class Designer](#)

C++ classes in Class Designer

10/21/2019 • 5 minutes to read • [Edit Online](#)

Class Designer supports C++ classes and visualizes native C++ classes in the same way as Visual Basic and C# class shapes, except that C++ classes can have multiple inheritance relationships. You can expand the class shape to show more fields and methods in the class or collapse it to conserve space.

NOTE

Class Designer does not support unions (a special type of class in which the memory allocated is only the amount necessary for the union's largest data member).

Simple inheritance

When you drag more than one class onto a class diagram, and the classes have a class inheritance relationship, an arrow connects them. The arrow points in the direction of the base class. For example, when the following classes are displayed in a class diagram, an arrow connects them, pointing from B to A:

```
class A {};
class B : A {};
```

You can also drag only class B to the class diagram, right-click the class shape for B, and then click **Show Base Classes**. This displays its base class: A.

Multiple inheritance

Class Designer supports the visualization of multiple-class inheritance relationships. *Multiple inheritance* is used when a derived class has attributes of more than one base class. Following is an example of multiple inheritance:

```
class Bird {};
class Swimmer {};
class Penguin : public Bird, public Swimmer {};
```

When you drag more than one class onto the class diagram, and the classes have a multiple-class inheritance relationship, an arrow connects them. The arrow points in the direction of the base classes.

Right-clicking a class shape and then clicking **Show Base Classes** displays the base classes for the selected class.

NOTE

The **Show Derived Classes** command is not supported for C++ code. You can display derived classes by going to **Class View**, expanding the type node, expanding the **Derived Types** subfolder, and then dragging those types onto the class diagram.

For more information about multiple-class inheritance, see [Multiple Inheritance](#) and [Multiple Base Classes](#).

Abstract classes

Class Designer supports abstract classes (also named "abstract base classes"). These are classes that you never

instantiate, but from which you can derive other classes. Using an example from "Multiple Inheritance" earlier in this document, you might instantiate the `Bird` class as individual objects as follows:

```
int main()
{
    Bird sparrow;
    Bird crow;
    Bird eagle;
}
```

However, you might not intend to instantiate the `Swimmer` class as individual objects. You might intend only to derive other types of animal classes from it, for example, `Penguin`, `Whale`, and `Fish`. In that case, you would declare the `Swimmer` class as an abstract base class.

To declare a class as abstract, you can use the `abstract` keyword. Members marked as abstract, or included in an abstract class, are virtual and must be implemented by classes that derive from the abstract class.

```
class Swimmer abstract
{
    virtual void swim();
    void dive();
};
```

You can also declare a class as abstract by including at least one pure virtual function:

```
class Swimmer
{
    virtual void swim() = 0;
    void dive();
};
```

When you display these declarations in a Class Diagram, the class name `Swimmer` and its pure virtual function `swim` are displayed in italic in an abstract class shape, together with the notation **Abstract Class**. Notice that the abstract class type shape is the same as that of a regular class, except that its border is a dotted line.

A class derived from an abstract base class must override each pure virtual function in the base class, or the derived class cannot be instantiated. So, for example, if you derive a `Fish` class from the `Swimmer` class, `Fish` must override the `swim` method:

```
class Fish : public Swimmer
{
    void swim(int speed);
};

int main()
{
    Fish guppy;
}
```

When you display this code in a Class Diagram, **Class Designer** draws an inheritance line from `Fish` to `Swimmer`.

Anonymous classes

Class Designer supports anonymous classes. *Anonymous class types* are classes declared without an identifier. They cannot have a constructor or destructor, cannot be passed as arguments to functions, and cannot be returned as return values from functions. You can use an anonymous class to replace a class name with a `typedef` name, as

in the following example:

```
typedef struct
{
    unsigned x;
    unsigned y;
} POINT;
```

Structures can also be anonymous. **Class Designer** displays anonymous classes and structures the same as it displays the respective type. Although you can declare and display anonymous classes and structures, **Class Designer** will not use the tag name that you specify. It will use the name that Class View generates. The class or structure appears in Class View and **Class Designer** as an element called **_unnamed**.

For more information about anonymous classes, see [Anonymous Class Types](#).

Template classes

Class Designer supports the visualization of template classes. Nested declarations are supported. The following table shows some typical declarations.

CODE ELEMENT	CLASS DESIGNER VIEW
template <class T>	A<T>
class A {};	Template Class
template <class T, class U>	A<T, U>
class A {};	Template Class
template <class T, int i>	A<T, i>
class A {};	Template Class
template <class T, template <class K> class U>	A<T, U>
class A {};	Template Class

The following table shows some examples of partial specialization.

CODE ELEMENT	CLASS DESIGNER VIEW
template<class T, class U>	A<T, U>
class A {};	Template Class
template<class T>	A<T, T>
class A<T, T> {};	Template Class
template <class T>	A<T, int>
class A<T, int> {};	Template Class

CODE ELEMENT	CLASS DESIGNER VIEW
template <class T1, class T2>	A<T1*, T2*>
class A<T1*, T2*> {};	Template Class

The following table shows some examples of inheritance in partial specialization.

CODE ELEMENT	CLASS DESIGNER VIEW
template <class T, class U>	A<T, U>
class A {};	Template Class
template <class TC>	B
class A<T, int> {};	Class
class B : A<int, float>	(points to Class A)
{};	C
class C : A<int, int>	Class
{};	(points to Class A)

The following table shows some examples of partial specialization template functions.

CODE ELEMENT	CLASS DESIGNER VIEW
class A	A
{	func<T, U> (+ 1 overload)
template <class T, class U>	
void func(T a, U b);	
template <class T>	
void func(T a, int b);	
};	
template <class T1>	A<T1>
class A {	Template Class
template <class T2>	B<T2>
class B {};	Template Class
};	(B is contained within class A under Nested Types)
template<> template<>	
class A<type>::B<type> {};	

CODE ELEMENT	CLASS DESIGNER VIEW
template <class T>	A
class C {};	Class
class A : C<int> {};	-> C<int>
	C<T>
	Template Class

The following table shows some examples of template inheritance.

CODE ELEMENT	CLASS DESIGNER VIEW
template <class T>	A
class C {};	Class
template<>	->B
class C<int> {	C<int>
class B {};	Class
}	(B is contained within class C under Nested Types)
class A : C<int>::B {};	C<T>
	Template Class

The following table shows some examples of canonical specialized class connection.

CODE ELEMENT	CLASS DESIGNER VIEW
template <class T>	A
class C {};	Class
template<>	->C<int>
class C<int> {};	C<int>
class A : C<int> {};	Class
class D : C<float> {};	C<T>
	Template Class
	D
	Class
	->C<float>

CODE ELEMENT	CLASS DESIGNER VIEW
<pre>class B { template <class T> T min (const T &a, const T &b); };</pre>	 min <T>

See also

- [Working with C++ Code](#)
- [Classes and Structs](#)
- [Anonymous Class Types](#)
- [Multiple Inheritance](#)
- [Multiple Base Classes](#)
- [Templates](#)

C++ structures in Class Designer

10/21/2019 • 2 minutes to read • [Edit Online](#)

Class Designer supports C++ structures, which are declared with the keyword `struct`. Following is an example:

```
struct MyStructure
{
    char a;
    int i;
    long j;
};
```

For more information about using the `struct` type, see [struct](#).

A C++ structure shape in a class diagram looks and works like a class shape, except that the label reads **Struct** and it has square corners instead of rounded corners.

CODE ELEMENT	CLASS DESIGNER VIEW
<code>struct StructureName {};</code>	StructureName Struct

See also

- [Working with C++ Code](#)
- [Classes and Structs](#)
- [struct](#)

C++ enumerations in Class Designer

10/21/2019 • 2 minutes to read • [Edit Online](#)

Class Designer supports C++ `enum` and scoped `enum class` types. Following is an example:

```
enum CardSuit {  
    Diamonds = 1,  
    Hearts = 2,  
    Clubs = 3,  
    Spades = 4  
};  
  
// or...  
enum class CardSuit {  
    Diamonds = 1,  
    Hearts = 2,  
    Clubs = 3,  
    Spades = 4  
};
```

A C++ enumeration shape in a class diagram looks and works like a structure shape, except that the label reads **Enum** or **Enum class**, it is pink instead of blue, and it has a colored border on the left and top margins. Both enumeration shapes and structure shapes have square corners.

For more information about using the `enum` type, see [Enumerations](#).

See also

- [Working with C++ Code](#)
- [Enumerations](#)

C++ typedefs in Class Designer

10/21/2019 • 2 minutes to read • [Edit Online](#)

Typedef statements create one or more layers of indirection between a name and its underlying type. **Class Designer** supports C++ typedef types, which are declared with the keyword `typedef`, for example:

```
typedef class coord
{
    void P(x,y);
    unsigned x;
    unsigned y;
} COORD;
```

You can then use this type to declare an instance:

```
COORD OriginPoint;
```

Class and struct shapes

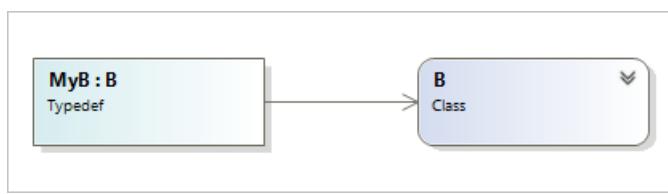
In **Class Designer**, a C++ typedef has the shape of the type specified in the typedef. If the source declares `typedef class`, the shape has rounded corners and the label **Class**. For `typedef struct`, the shape has square corners and the label **Struct**.

Classes and structures can have nested typedefs declared within them. In **Class Designer**, class and structure shapes can show nested typedef declarations as nested shapes.

Typedef shapes support the **Show as Association** and **Show as Collection Association** commands on the right-click menu (context menu).

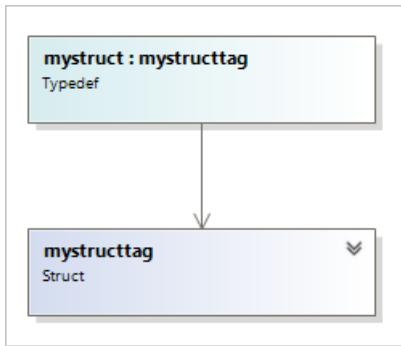
Class typedef example

```
class B {};
typedef B MyB;
```



Struct typedef example

```
typedef struct mystructtag
{
    int i;
    double f;
} mystruct;
```



Unnamed typedefs

Although you can declare a typedef without a name, **Class Designer** doesn't use the tag name that you specify. **Class Designer** uses the name that **Class View** generates. For example, the following declaration is valid, but it appears in **Class View** and **Class Designer** as an object named `_unnamed`:

```
typedef class coord
{
    void P(x,y);
    unsigned x;
    unsigned y;
};
```

NOTE

Class Designer does not display typedefs whose source type is a function pointer.

See also

- [Work with C++ Code](#)
- [Typealiases](#)

Keyboard and mouse shortcuts in the Class Diagram and Class Details window

10/31/2019 • 5 minutes to read • [Edit Online](#)

You can use the keyboard in addition to the mouse to perform navigational actions in **Class Designer** and in the **Class Details** window.

Use the mouse in Class Designer

The following mouse actions are supported in class diagrams:

MOUSE COMBINATION	CONTEXT	DESCRIPTION
Double-click	Shape elements	Opens the code editor.
Double-click	Lollipop connector	Expand/collapse lollipop.
Double-click	Lollipop connector label	Invokes Show Interface command.
Mouse Wheel	Class diagram	Scroll vertically.
Shift + Mouse Wheel	Class diagram	Scroll horizontally.
Ctrl + Mouse Wheel	Class diagram	Zoom.
Ctrl+Shift + click	Class diagram	Zoom.

Use the mouse in the Class Details window

Using a mouse, you can change the appearance of the **Class Details** window and the data it displays in the following ways:

- Clicking any editable cell lets you edit the contents of that cell. Your changes are reflected in all places that data is stored or displayed, including in the **Properties** window and in source code.
- Clicking any cell of a row causes the **Properties** window to display the properties for the element represented by that row.
- To change the width of a column, drag the boundary on the right side of the column heading until the column is the width you want.
- You can expand or collapse compartment or property nodes by clicking the arrow symbols to the left of the row.
- The **Class Details** window offers several buttons for creating new members in the current class and for navigating among the members' compartments in the **Class Details** window grid.

Use the keyboard in Class Designer

The following keyboard actions are supported in class diagrams:

KEY	CONTEXT	DESCRIPTION
Arrow keys	Inside type shapes	Tree-style navigation on shape contents (wrapping around shape is supported). Left and right keys expand/collapse current item if it is expandable and navigate to parent if not (see tree-view navigation for detailed behavior).
Arrow keys	Top-level shapes	Moving shapes on the diagram.
Shift+arrow keys	Inside type shapes	Building continuous selection consisting of shape elements such as members, nested types, or compartments. These shortcuts do not support wrapping around.
Home	Inside type shapes	Navigate to the top-level shape title.
Home	Top-level shapes	Navigate to first shape on the diagram.
End	Inside type shapes	Navigate to last visible element inside the shape.
End	Top-level shapes	Navigate to the last shape on the diagram.
Shift+ Home	Inside type shape	Selects elements within the shape starting with the current item and ending with the top-most item on the same shape.
Shift+ End	Inside type shape	Same as Shift+ Home but in top-down direction.
Enter	All contexts	Invokes default action on the shape which is also available via double-click. In most cases this is View Code but some elements define it differently (lollipops, compartment headers, lollipop labels).
+ and -	All contexts	If currently focused element is expandable, these keys expand or collapse the element.
>	All contexts	On elements with children, this expands the element if it is collapsed and navigates to first child.
<	All contexts	Navigates to the parent element.
Alt+Shift+L	Inside type shapes + on type shapes.	Navigates to the lollipop of currently selected shape if it is present.

KEY	CONTEXT	DESCRIPTION
Alt+Shift+B	Inside type shapes + on type shapes.	If base type list is shown on the type shape and has more than one item, this toggles expansion state of the list (collapse/expand).
Delete	On type and comment shapes	Invokes Remove from Diagram command.
Delete	On everything else.	Invokes Delete from Code command (members, parameters, associations, inheritance, lollipop labels).
Ctrl+Delete	All contexts	Invokes Delete from Code command on selection.
Tab	All contexts	Navigates to next child within the same parent (supports wrapping).
Shift+Tab	All contexts	Navigates to previous child within the same parent (supports wrapping).
Spacebar	All contexts	Toggles selection on the current element.

Use the keyboard in the Class Details window

NOTE

The following key bindings were chosen to specifically mimic the experience of typing code.

Use the following keys to navigate the **Class Details** window:

Key	Result
,	If the cursor is in a parameter row, typing a comma moves the cursor to the Name field of the next parameter. If the cursor is in the last parameter row of a method, it moves the cursor to the <add parameter> field, which you can use to create a new parameter. If the cursor is elsewhere in the Class Details window, typing a comma literally adds a comma in the current field.
;(semicolon) or) (closing parenthesis)	Move the cursor to the Name field of the next member row in the Class Details window grid.

Tab	Moves the cursor to the next field, first moving left to right and then top to bottom. If the cursor is moving from a field in which you have typed text, Class Details processes that text and stores it if it does not produce an error. If the cursor is on an empty field such as <add parameter>, Tab moves it to the first field of the next row.
Spacebar	Moves the cursor to the next field, first moving left to right and then top to bottom. If the cursor is on an empty field such as <add parameter>, it moves to the first field of the next row. Note that <space> typed immediately after a comma is ignored. If the cursor is in the Summary field, typing a space adds a space character. If the cursor is in the Hide column of a given row, typing a space toggles the value of the Hide checkbox.
Ctrl+Tab	Switch to another document window. For example, switch from the Class Details window to an open code file.
Esc	If you have begun to type text in a field, pressing ESC acts as an undo key, reverting the field's contents to its previous value. If the Class Details Window has general focus, but no specific cell has focus, pressing ESC moves focus away from the Class Details window.
Up arrow and down arrow	These keys move the cursor from row to row vertically in the Class Details window grid.
Left arrow	If the cursor is in the Name column, pressing the left arrow collapses the current node in the hierarchy (if it is open).
Right arrow	If the cursor is in the Name column, pressing the right arrow expands the current node in the hierarchy (if it is collapsed).

See also

- [Create and configure type members](#)
- [How to use the keyboard exclusively](#)
- [Default keyboard shortcuts in Visual Studio](#)
- [Keyboard shortcuts in Blend](#)

Class Designer errors

10/25/2019 • 2 minutes to read • [Edit Online](#)

Class Designer does not track the location of your source files, so modifying your project structure or moving source files in the project can cause **Class Designer** to lose track of the type. For example, it's common to modify the source type of a typedef, base classes, and association types. You might receive an error such as **Class Designer is unable to display this type**. To resolve the error, drag the modified or relocated source code to the class diagram again to display it.

Resources

You can find assistance with other errors and warnings in the following resources:

- [Work with Visual C++ code](#) includes troubleshooting information about displaying C++ in a class diagram.
- [Visual Studio Class Designer forum](#) provides a forum for questions about **Class Designer**.

See also

- [Design and view classes and types](#)

Make code work in Visual Studio

10/18/2019 • 10 minutes to read • [Edit Online](#)

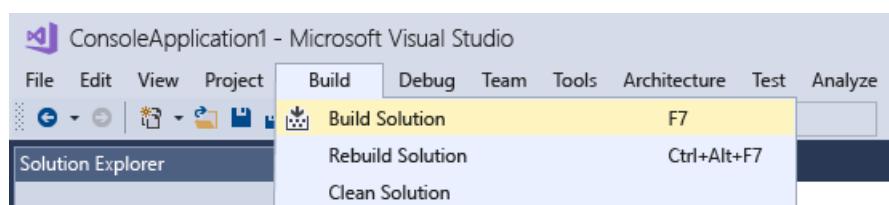
Visual Studio provides a powerful integrated set of project build and debugging tools. In this article, find out how Visual Studio can help you find problems in your code using build output, code analysis, debugging tools, and unit tests.

You've figured out the editor and created some code. Now, you want to make sure the code works properly. In Visual Studio, as with most IDEs, there are two phases to making code work: building the code to catch and resolve project and compiler errors, and running the code to find run-time and dynamic errors.

Build your code

There are two basic types of build configuration: **Debug** and **Release**. The **Debug** configuration produces a slower, larger executable that allows for a richer interactive run-time debugging experience. The **Debug** executable should never be shipped. The **Release** configuration builds a faster, optimized executable that's appropriate to ship (at least from the perspective of the compiler). The default build configuration is **Debug**.

The easiest way to build your project is to press **F7**, but you can also start the build by selecting **Build > Build Solution** from the main menu.



You can observe the build process in the **Output** window at the bottom of the Visual Studio UI. Errors, warnings, and build operations are displayed here. If you have errors (or if you have warnings above a configured level), your build fails. You can click on the errors and warnings to go to the line where they occurred. Rebuild your project by either pressing **F7** again (to recompile only the files with errors) or **Ctrl+Alt+F7** (for a clean and complete rebuild).

There are two tabbed windows in the results window below the editor: the **Output** window, which contains the raw compiler output (including error messages); and the **Error List** window, which provides a sortable and filterable list of all errors and warnings.

When build succeeds, you see results like this in the **Output** window:

```
Output
Show output from: Build
1>----- Build started: Project: MyNewApp, Configuration: Release Any CPU -----
1>  MyNewApp -> c:\users\user1\documents\visual studio 2015\Projects\MyNewApp\MyNewApp\bin\Release\
      MyNewApp.exe
===== Build: 1 succeeded, 0 failed, 0 up-to-date, 0 skipped ======
```

A screenshot of the 'Output' window in Visual Studio. The window title is 'Output'. A dropdown menu 'Show output from:' is set to 'Build'. The main pane displays the build log for a project named 'MyNewApp' in 'Release' configuration. The log shows the build starting, the project being built, and the resulting executable 'MyNewApp.exe'. At the bottom, a summary states 'Build: 1 succeeded, 0 failed, 0 up-to-date, 0 skipped'.

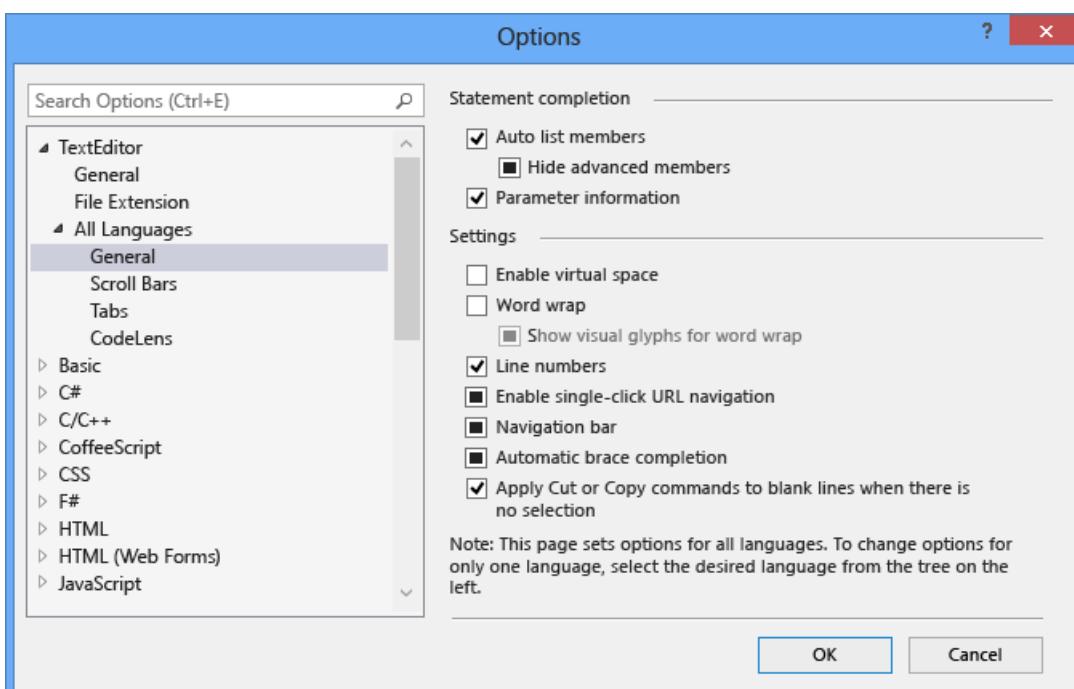
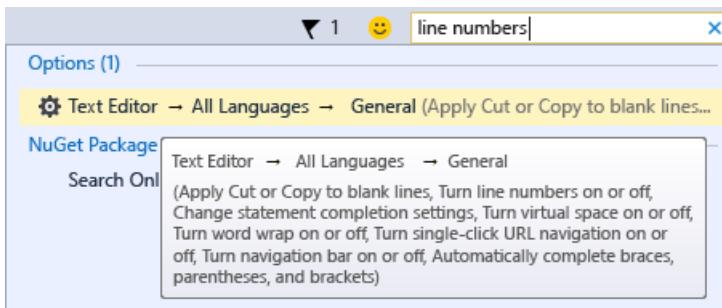
Review the Error List

Unless you've made no modifications to code you've previously and successfully compiled, you probably have an error. If you're new to coding, you probably have lots of them. Errors are sometimes obvious, such as a simple syntax error or incorrect variable name, and sometimes they are difficult to understand, with only a cryptic code to guide you. For a cleaner view of the issues, navigate to the bottom of the build **Output** window, and click the **Error List** tab. This takes you to a more organized view of the errors and warnings for your project, and gives you some extra options as well.

Error List			
	Code	Description	Project
✖	CS1002	; expected	ConsoleApplication1
⚠	CS0219	The variable 'analyzeThis' is assigned but its value is never used	ConsoleApplication1
✖	CS0818	Implicitly-typed variables must be initialized	ConsoleApplication1
⚠	CS0168	The variable 'iMadeAnError' is declared but never used	ConsoleApplication1

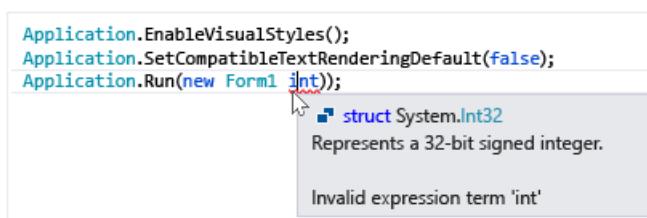
Output Error List

Click on the error line in the **Error List** window to jump to the line the error occurs in. (Or turn on line numbers by pressing **Ctrl+Q**, typing **line numbers**, and then choosing **Turn line numbers on or off** from the results. This is the fastest way to get to the **Options** dialog where you can turn on line numbers.)

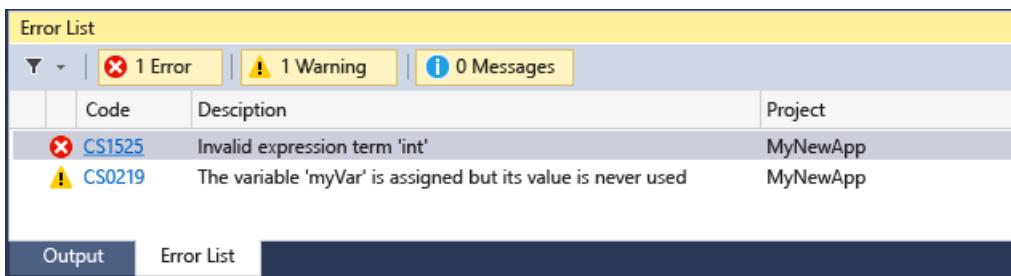


Press **Ctrl+G** to quickly jump to the line number where the error occurred.

The error is identified by a red "squiggle" underscore. Hover over it for additional details. Make the fix and it will go away, although you may introduce a new error with the correction. (This is called a "regression".)

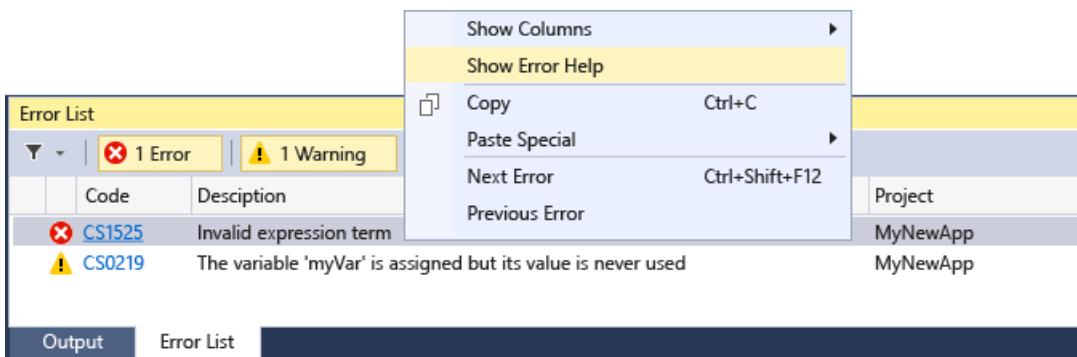


Walk through the error list and address all the errors in your code.



Review errors in detail

Many errors may make no sense to you, phrased as they are in the terms of the compiler. In those cases, you'll need additional information. From the **Error List** window, you can do an automatic Bing search for more information on the error or warning. Right-click on the corresponding entry line and select **Show Error Help** from the context menu, or click on the hyperlinked error code value in the **Code** column of the **Error List**.



Depending on your settings, either your web browser displays the search results for the error code and text, or a tab opens inside Visual Studio and shows the results of the Bing search. The results are from many different sources on the Internet, and not all may be helpful.

Use code analysis

Code analyzers look for common code problems that can lead to run-time errors or problems in code management.

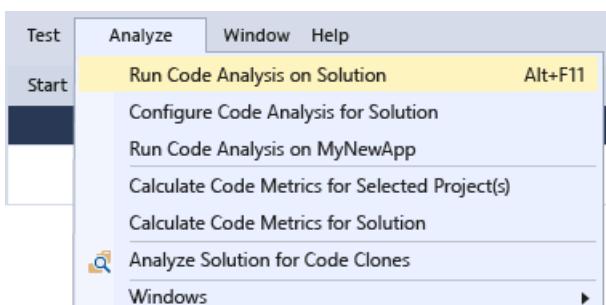
C# and Visual Basic code analysis

Visual Studio includes a built-in set of [.NET Compiler Platform analyzers](#) that examine C# and Visual Basic code as you type. You can install additional analyzers as a Visual Studio extension, or as a NuGet package. If rule violations are found, they are reported both in the Error List and in the code editor as a squiggle under the offending code.

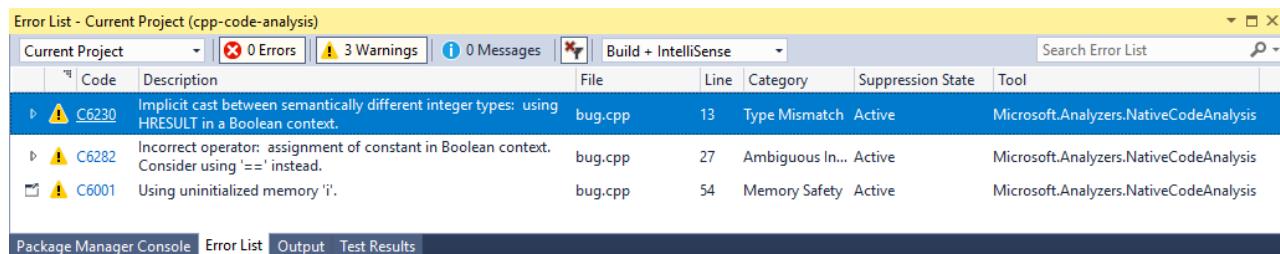
C++ code analysis

To analyze C++ code, run [static code analysis](#). Get in the habit of running it once you've cleaned up the obvious errors that prevent a successful build, and take some time to address the warnings it may produce. You'll save yourself some headaches down the road, and you may learn a few code style techniques.

Press **Alt+F11** (or select **Analyze > Run Code Analysis on Solution** from the top menu) to start static code analysis.

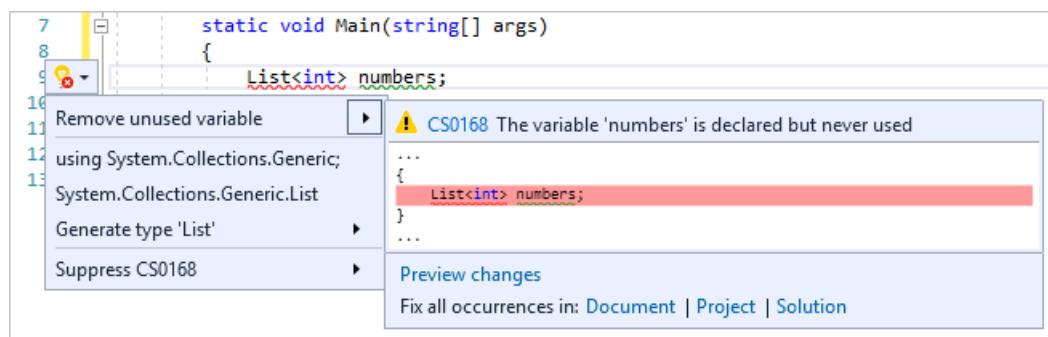


Any new or updated warnings appear in the **Error List** tab at the bottom of the IDE. Click on the warnings to jump to them in code.

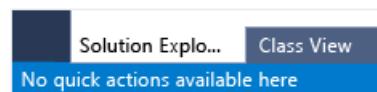


Use Quick Actions to fix or refactor code

Quick Actions, available from the light bulb or screwdriver icon, let you refactor code inline. They are an easy way to fix common warnings quickly and effectively in C#, C++, and Visual Basic code. To access them, right-click on a warning squiggle and select **Quick Actions and refactorings**. Or, when your cursor is on the line with the colored squiggle, press **Ctrl+.** or select the light bulb, error light bulb, or screwdriver icon in the margin. You'll see a list of possible fixes or refactorings you can apply to that line of code.



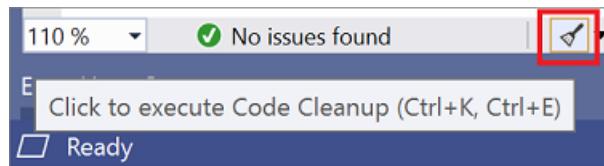
Quick Actions can be used wherever code analyzers determine there's an opportunity to fix, refactor, or improve your code. Click on any line of code, right-click to open the context menu, and select **Quick Actions and refactorings**. If refactoring or improvement options are available, they are displayed. Otherwise, the message **No quick actions available here** displays in the lower-left corner of the IDE.



With experience, you can quickly use the arrow keys and **Ctrl+.** to check for easy refactoring opportunities and clean up your code!

Run Code Cleanup

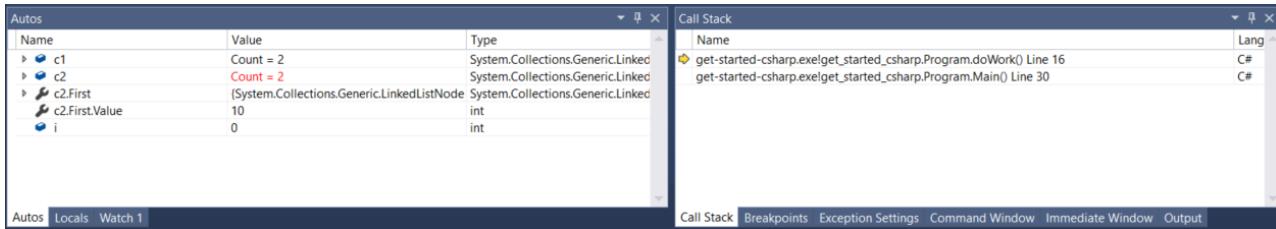
Visual Studio provides [on-demand formatting of your C# code file](#), including code style preferences, through the **Code Cleanup** button at the bottom of the editor.



In addition to formatting your file for spaces, indents, et cetera, **Code Cleanup** also applies a set of code style conventions that you define. Your preferences for each code style are read from the [EditorConfig file](#), if you have one for the project, or from the [code style settings](#) in the **Options** dialog box.

Debug your running code

Now that you've successfully built your code and performed a little clean up, run it by pressing **F5** or selecting **Debug > Start Debugging**. This starts your app in a debug environment so you can observe its behavior in detail. The Visual Studio IDE changes while your app is running: the **Output** window is replaced by two new ones (in the default window configuration), the **Autos/Locals/Watch** tabbed window and the **Call Stack/Breakpoints/Exception Settings/Output** tabbed window. These windows have multiple tabs that allow you to inspect and evaluate your app's variables, threads, call stacks, and various other behaviors as it runs.



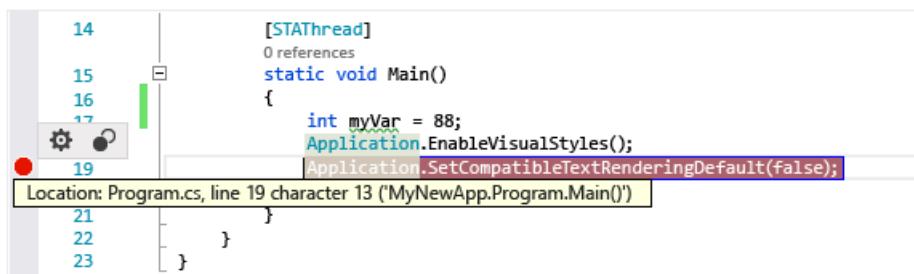
Stop your app by pressing **Shift+F5** or by clicking the **Stop** button. Or, you can just close the app's main window (or command-line dialog).

If your code ran perfectly and exactly as expected, congratulations! However, if it hung, or crashed, or gave you some strange results, you'll need to find the source of those problems and fix the bugs.

Set simple breakpoints

Breakpoints are the most basic and essential feature of reliable debugging. A breakpoint indicates where Visual Studio should suspend your running code so you can take a look at the values of variables, or the behavior of memory, or whether or not a branch of code is getting run. You don't need to rebuild a project after setting and removing breakpoints.

Set a breakpoint by clicking in the far margin of the line where you want the break to occur, or press **F9** to set a breakpoint on the current line of code. When you run your code, it will pause (or *break*) before the instructions for this line of code are executed.



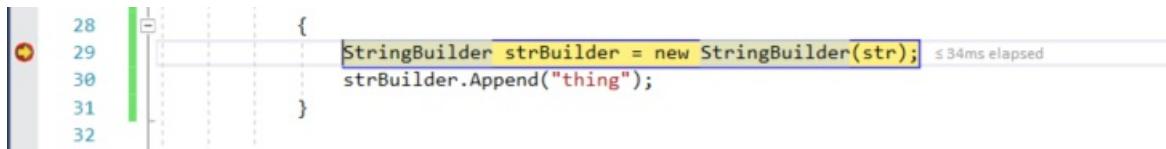
Common uses for breakpoints include:

- To narrow down the source of a crash or hang, scatter breakpoints throughout and around the code of the method call you think is causing the failure. As you run code in the debugger, remove and then reset the breakpoints closer together until you find the offending line of code. See the next section to learn how to run code in the debugger.
- When you introduce new code, set a breakpoint at the beginning of it, and run the code to make sure it is behaving as expected.
- If you've implemented a complicated behavior, set breakpoints for the algorithmic code so you can inspect the values of the variables and data when the program breaks.
- If you're writing C or C++ code, use breakpoints to stop the code so you can inspect address values (look for NULL) and reference counts when debugging for memory-related failures.

For more information about using breakpoints, read [Using breakpoints](#).

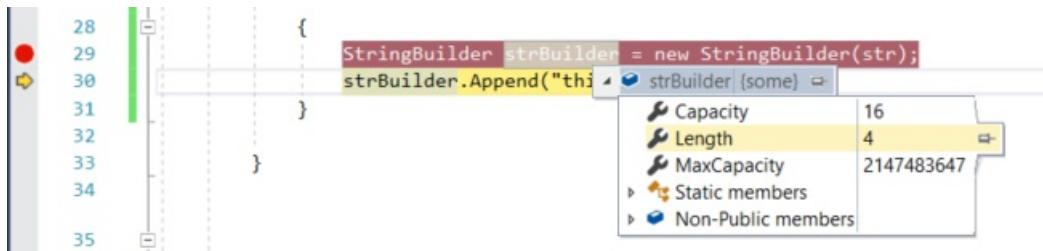
Inspect your code at run-time

When your running code hits a breakpoint and pauses, the line of code marked in yellow (the current statement) has not executed yet. At this point, you may want to execute the current statement and then inspect the changed values. You can use several *step* commands to execute code in the debugger. If the marked code is a method call, you can step into it by pressing **F11**. You can also *step over* the line of code by pressing **F10**. For additional commands and details on how to step through code, read [Navigate code with the debugger](#).



In the preceding illustration, you can advance the debugger one statement by pressing either **F10** or **F11** (since there is no method call here, both commands have the same result).

While the debugger is paused, you can inspect your variables and call stacks to determine what is going on. Are the values in the ranges you expect to see? Are calls being made in the right order?



Hover over a variable to see its current value and references. If you see a value you didn't expect, you probably have a bug in the preceding or calling code. For more in-depth debugging information, [learn more](#) about using the debugger.

Additionally, Visual Studio displays the **Diagnostic Tools** window, where you can observe your app's CPU and memory usage over time. Later in your app development, you can use these tools to look for unanticipated heavy CPU usage or memory allocation. Use it in conjunction with the **Watch** window and breakpoints to determine what's causing unexpected heavy usage or unreleased resources. For more information, see [Profiling feature tour](#).

Run unit tests

Unit tests are your first line of defense against code bugs because, when done correctly, they test a single "unit" of code, typically a single function, and are easier to debug than your full program. Visual Studio installs the Microsoft unit testing frameworks for both managed and native code. Use a unit testing framework to create unit tests, run them, and report the results of these tests. Rerun unit tests when you make changes, to test that your code is still working correctly. With Visual Studio Enterprise edition, you can run tests automatically after every build.

To get started, read [Generate unit tests for your code with IntelliTest](#).

To learn more about unit tests in Visual Studio and how they can help you create better quality code, read [Unit test basics](#).

See also

- [First look at the debugger](#)
- [Learn more about using the debugger](#)
- [Generate and fix code](#)

Productivity tips for Visual Studio

10/18/2019 • 8 minutes to read • [Edit Online](#)

This article discusses tips for Visual Studio features that help you write, navigate, and debug your code more quickly and efficiently.

For information about helpful keyboard shortcuts, see [Productivity shortcuts](#). For a complete list of command shortcuts, see [Default keyboard shortcuts](#).

Write code

Write code more quickly by using the following features.

- **Use convenience commands.** Visual Studio contains various commands to help you accomplish common editing tasks faster. For example, you can choose a command to easily duplicate a line of code without having to copy it, reposition the cursor, and then paste it. Choose **Edit > Duplicate** or press **Ctrl+E,V**. You can also quickly expand or contract a selection of text by choosing **Edit > Advanced > Expand Selection** or **Edit > Advanced > Contract Selection**, or by pressing **Shift+Alt+=** or **Shift+Alt+-**.
- **Use IntelliSense.** As you enter code in the editor, IntelliSense information, such as List Members, Parameter Info, Quick Info, Signature Help, and Complete Word, appears. These features support fuzzy matching of text; for example, the results lists for List Members includes not only entries that start with the characters that you have entered but also entries that contain the character combination anywhere in their names. For more information, see [Use IntelliSense](#).
- **Change auto-insertion of IntelliSense options as you enter code.** By switching IntelliSense to suggestion mode, you can specify that IntelliSense options are inserted only if you explicitly choose them.

To enable suggestion mode, choose the **Ctrl+Alt+Spacebar** keys, or, on the menu bar, choose **Edit > IntelliSense > Toggle Completion Mode**.

- **Use code snippets.** You can use built-in snippets, or create your own snippets.

To insert a snippet, on the menu bar, choose **Edit > IntelliSense > Insert Snippet** or **Surround With**, or open the shortcut menu in a file and choose **Snippet > Insert Snippet** or **Surround With**. For more information, see [Code Snippets](#).

- **Fix code errors inline.** Quick Actions let you easily refactor, generate, or otherwise modify code with a single action. These actions can be applied using the screwdriver  or light bulb  icons, or by pressing **Alt+Enter** or **Ctrl+.** when your cursor is on the appropriate line of code. See [Quick Actions](#) for more information.
- **Show and edit the definition of a code element.** You can quickly show and edit the module in which a code element, such as a member, a variable, or a local, is defined.

To open a definition in a pop-up window, highlight the element and then choose the **Alt+F12** keys, or open the shortcut menu for the element and then choose **Peek Definition**. To open a definition in a separate code window, open the shortcut menu for the element, and then choose **Go to Definition**.

- **Use sample applications.** You can speed up application development by downloading and installing sample applications from [Microsoft Developer Network](#). You can also learn a particular technology or programming concept by downloading and exploring a Sample Pack for that area.

Navigate within your code

You can use various techniques to find and move to specific locations in your code more quickly.

- **Bookmark lines of code.** You can use bookmarks to navigate quickly to specific lines of code in a file.

To set a bookmark, on the menu bar, choose **Edit > Bookmarks > Toggle Bookmark**. You can view all of the bookmarks for a solution in the **Bookmarks** window. For more information, see [Set bookmarks in code](#).

- **Search for symbol definitions in a file.** You can search within a solution to locate symbol definitions and file names, but search results don't include namespaces or local variables.

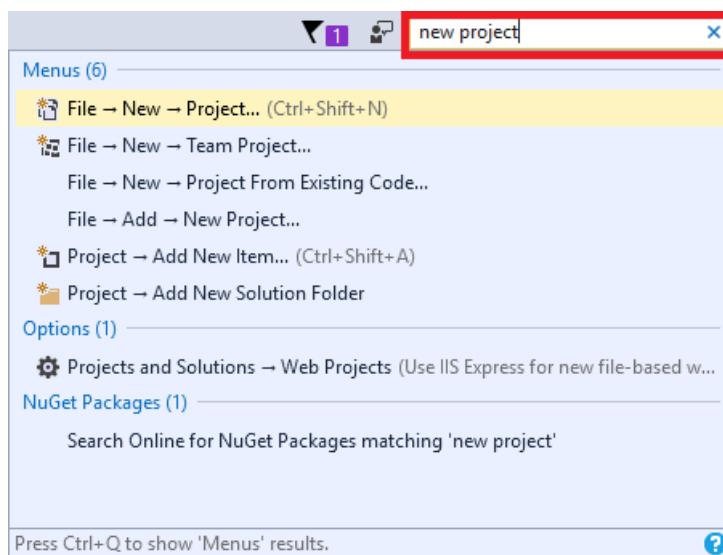
To access this feature, on the menu bar, choose **Edit > Navigate To**.

- **Browse the overall structure of your code.** In **Solution Explorer**, you can search and browse classes and their types and members in your projects. You can also search for symbols, view a method's Call Hierarchy, find symbol references, and perform other tasks. If you choose a code element in **Solution Explorer**, the associated file opens in a **Preview** tab, and the cursor moves to the element in the file. For more information, see [View the structure of code](#).

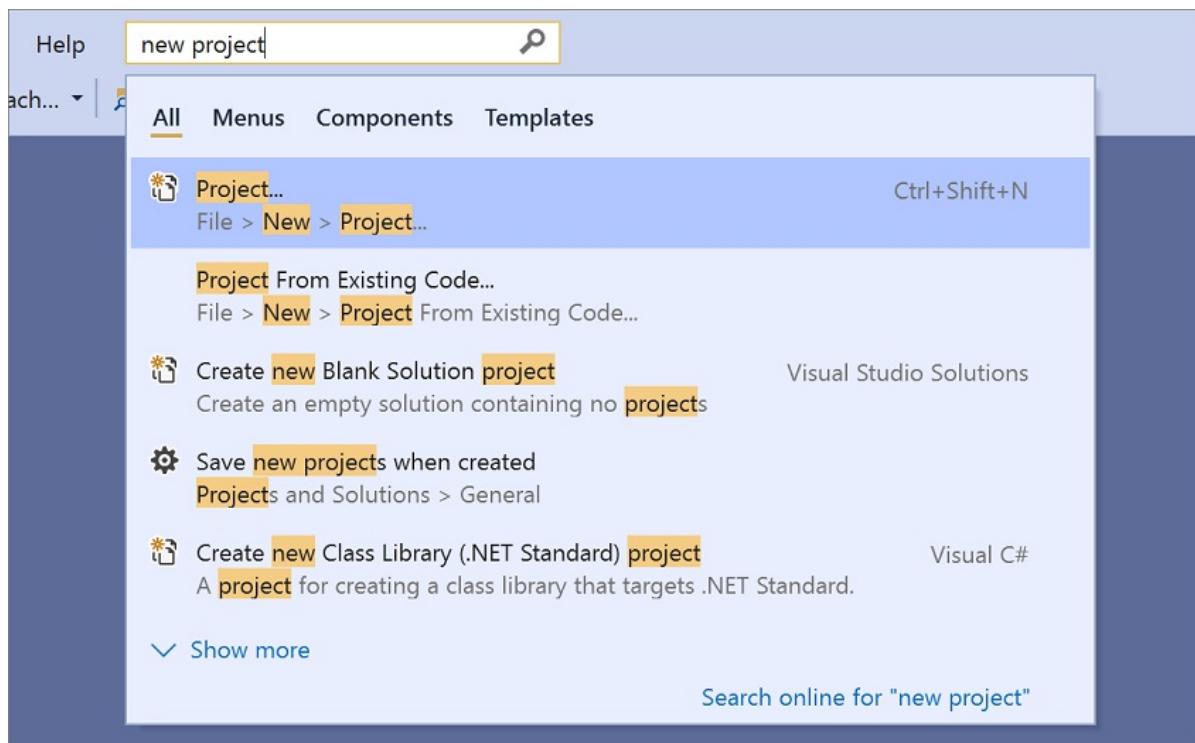
Find items faster

You can search across the IDE for commands, files, and options, in addition to filtering the contents of tool windows to show only relevant information for your current task.

- **Filter the contents of tool windows.** You can search within the contents of many tool windows, such as the **Toolbox**, the **Properties** window, and **Solution Explorer**, but display only items whose names contain the characters that you specify.
- **Display only the errors you want to address.** If you choose the **Filter** button on the **Error List** toolbar, you can reduce the number of errors that appear in the **Error List** window. You can display only the errors in the files that are open in the editor, only the errors in the current file, or only the errors in the current project. You can also search within the **Error List** window to find specific errors.
- **Find dialog boxes, menu commands, options, and more.** In the search box, enter keywords or phrases for the items that you're trying to find. For example, the following options appear if you enter **new project**:



Quick Launch displays links to create a new project, to add a new item to a project, and to the **Projects and Solutions** page in the **Options** dialog box, among others. Search results can also include project files and tool windows.

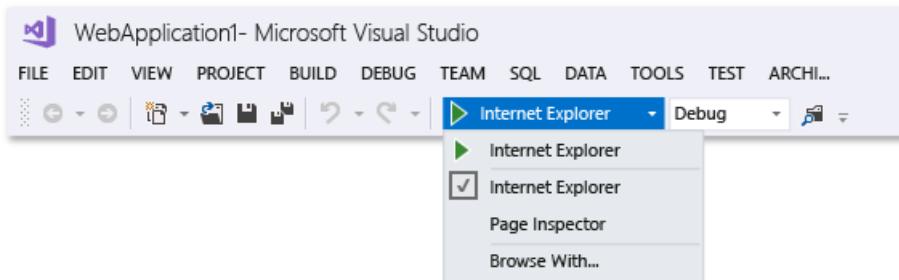


Press **Ctrl+Q** to jump straight to the search box.

Debug code

Debugging can consume a lot of time, but the following tips can help you speed up the process.

- **Test the same page, application, or site in different browsers.** As you debug your code, you can easily switch among the installed web browsers, including [Page Inspector \(Visual Studio\)](#), without having to open the **Browse With** dialog box. You can use the **Debug Target** list, which is on the **Standard** toolbar next to the **Start Debugging** button, to quickly verify which browser you're using as you debug or view pages.



- **Set temporary breakpoints.** You can create a temporary breakpoint in the current line of code and start the debugger simultaneously. When you hit that line of code, the debugger enters break mode. For more information, see [Navigate through code with the debugger](#).

To use this feature, choose the **Ctrl+F10** keys, or open the shortcut menu for the line of code on which you want to break, and then choose **Run To Cursor**.

- **Move the execution point during debugging.** You can move the current execution point to a different section of code and then restart debugging from that point. This technique is useful if you want to debug a section of code without having to recreate all of the steps that are required to reach that section. For more information, see [Navigate through code with the debugger](#).

To move the execution point, drag the yellow arrowhead to a location where you want to set the next statement in the same source file, and then choose the **F5** key to continue debugging.

- **Capture value information for variables.** You can add a DataTip to a variable in your code and pin it so that you can access the last known value for the variable after debugging has finished. For more

information, see [View data values in Data Tips](#).

To add a DataTip, the debugger must be in break mode. Place the cursor on the variable, and then choose the pin button on the DataTip that appears. When debugging is stopped, a blue pin icon appears in the source file next to the line of code that contains the variable. If you point to the blue pin, the value of the variable from the most recent debugging session appears.

- **Clear the Immediate window.** You can erase the contents of the [Immediate window](#) at design time by entering `>cls` or `>Edit.ClearAll`

For more information about additional commands, see [Visual Studio command aliases](#).

Access Visual Studio tools

You can quickly access the Developer Command Prompt, or another Visual Studio tool, if you pin it to the Start menu or the taskbar.

1. In Windows Explorer, browse to `%ProgramData%\Microsoft\Windows\Start Menu\Programs\Visual Studio 2017\Visual Studio Tools`.
1. In Windows Explorer, browse to `%ProgramData%\Microsoft\Windows\Start Menu\Programs\Visual Studio 2019\Visual Studio Tools`.
2. Right-click or open the context menu for **Developer Command Prompt**, and then choose **Pin to Start** or **Pin to taskbar**.

Manage files, toolbars, and windows

At any one time, you may be working in multiple code files and moving among several tool windows as you develop an application. You can keep organized by using the following tips:

- **Keep files that you frequently use visible in the editor.** You can pin files to the left side of the tab well so that they remain visible regardless of how many files are open in the editor.

To pin a file, choose the file's tab, and then choose the **Toggle Pin Status** button.

- **Move documents and windows to other monitors.** If you use more than one monitor when you develop applications, you can work on portions of your application more easily by moving files that are open in the editor to another monitor. You can also move tool windows, such as debugger windows, to another monitor and tab dock document and tool windows together to create "rafts." For more information, see [Customize window layouts in Visual Studio](#).

You can also manage files more easily by creating another instance of **Solution Explorer** and moving it to another monitor. To create another instance of **Solution Explorer**, open a shortcut menu in **Solution Explorer**, and then choose **New Solution Explorer View**.

- **Customize the fonts that appear in Visual Studio.** You can change the font face, size, and color that's used for text in the IDE. For example, you can customize the color of specific code elements in the editor and the font face in tool windows or throughout the IDE. For more information, see [How to: Change fonts and colors](#) and [How to: Change fonts and colors in the editor](#).

See also

- [Visual Studio tips and tricks blog post](#)
- [Default keyboard shortcuts for frequently used commands](#)
- [How to: Customize menus and toolbars](#)
- [Walkthrough: Create a simple application](#)

- Accessibility tips and tricks

Shortcut tips for Visual Studio

10/18/2019 • 3 minutes to read • [Edit Online](#)

You can navigate in Visual Studio more easily by using the shortcuts in this article. These shortcuts include keyboard and mouse shortcuts as well as text you can enter to help accomplish a task more easily.

For a complete list of command shortcuts, see [Default keyboard shortcuts](#).

NOTE

This topic applies to Visual Studio on Windows. For Visual Studio for Mac, see [Common keyboard shortcuts in Visual Studio for Mac](#).

Window management

TASK	SHORTCUT
Maximize floating window	Double-click on title bar
Maximize/minimize windows	Win+ Up arrow / Win+ Down arrow
Redock floating window	Ctrl+double-click on title bar
Move/dock floating windows	Win+ Left arrow / Win+ Right arrow
Close active document	Ctrl+ F4
Show open file list	Ctrl+ Alt+ Down arrow
Show all floating windows	Ctrl+ Shift+ M
Show jump list	Win+ Alt+ N
Start new instance	Win+ Shift+ N
Switch between windows	Win+ N

Search

TASK	SHORTCUT
Solution Explorer search	Ctrl+;
Place focus in search box in any tool window (except editor)	Alt+` when the tool window has focus
Search Visual Studio	Ctrl+Q

TASK	SHORTCUT
Search box results filter	@opt - Options @cmd - Commands @mru - Most recently used @doc - Open documents
Search in Tools Options	Ctrl+E

Find in the editor

TASK	SHORTCUT
Quick Find	Ctrl+F
Quick Find Next Result	Enter
Quick Find Previous Result	Shift+Enter
Quick Find Expand Drop Down	Alt+Down arrow
Dismiss Find	Esc
Quick Replace	Ctrl+H
Quick Replace - Replace Next	Alt+R
Quick Replace - Replace All	Alt+A
Find in Files	Ctrl+Shift+F
Replace in Files	Ctrl+Shift+H

Code editor

TASK	SHORTCUT
Go To All	Ctrl+T
Go to recent files	Ctrl+T,R
Multi-caret multiple insertion points	Ctrl+Alt+click
Multi-caret add matching selection	Shift+Alt+Ins
Format Document	Ctrl+K,D
IntelliSense suggestion mode	Ctrl+Alt+Space (Toggle)
Force show IntelliSense	Ctrl+J

TASK	SHORTCUT
Quick Actions	Ctrl+.
Snippet picker	Ctrl+K,X or ?Tab (Visual Basic)
Surround With	Ctrl+KS
Show Quick Info	Ctrl+KI
Navigate To	Ctrl+,
Navigate highlighted references	Ctrl+Shift+Up (Previous) Ctrl+Shift+Down (Next)
Editor zoom	Ctrl+Shift+> (In) Ctrl+Shift+< (Out)
Block selection	Hold Alt and drag mouse Shift+Alt+Arrow keys
Move line up/down	Alt+Up arrow / Alt+Down arrow
Duplicate line	Ctrl+E,V
Expand selection	Shift+Alt+=
Contract selection	Shift+Alt+-
Go To Definition	F12
Peek Definition	Alt+F12
Go To Definition stack	Ctrl+Shift+8 (Back) Ctrl+Shift+7 (Forward)
Close the Peek Definition window	Esc
Promote the Peek Definition window to a regular document tab	Ctrl+Alt+Home
Navigate between multiple Peek Definition windows	Ctrl+Alt+- and Ctrl+Alt+=
Navigate between multiple Peek results	F8 and Shift+F8
Toggle between the code editor window and the Peek Definition window	Shift+Esc
Go to enclosing block	Ctrl+Alt+Up arrow
Go to next/previous issue	Alt+PgUp / Alt+PgDn
Contextual navigation menu	Alt+`

Toolbars

TASK	SHORTCUT
Add buttons	Select the toolbar overflow button
Find combo in standard toolbar	Ctrl+D
Find textbox command mode	Type >
Create new alias	Type alias <new alias> <command> in the Command window

Debugging

TASK	SHORTCUT
Start debugging	F5
Stop debugging	Shift+F5
Restart debugging	Ctrl+Shift+F5
Step over	F10
Step into	F11
Step out	Shift+F11
Run to cursor	Ctrl+F10
Set next statement	Ctrl+Shift+F10
Set and toggle breakpoint	F9
Disable breakpoint	Ctrl+F9
Immediate window	Ctrl+Alt+I
Immediate window command mode	Type >
Immediate window - clear buffer	Type cls
Immediate window - print value	Type ?varname

See also

- [Accessibility tips and tricks](#)
- [Productivity features in Visual Studio](#)
- [Default keyboard shortcuts](#)
- [Common keyboard shortcuts in Visual Studio for Mac](#)

Visual Studio productivity guide for C# developers

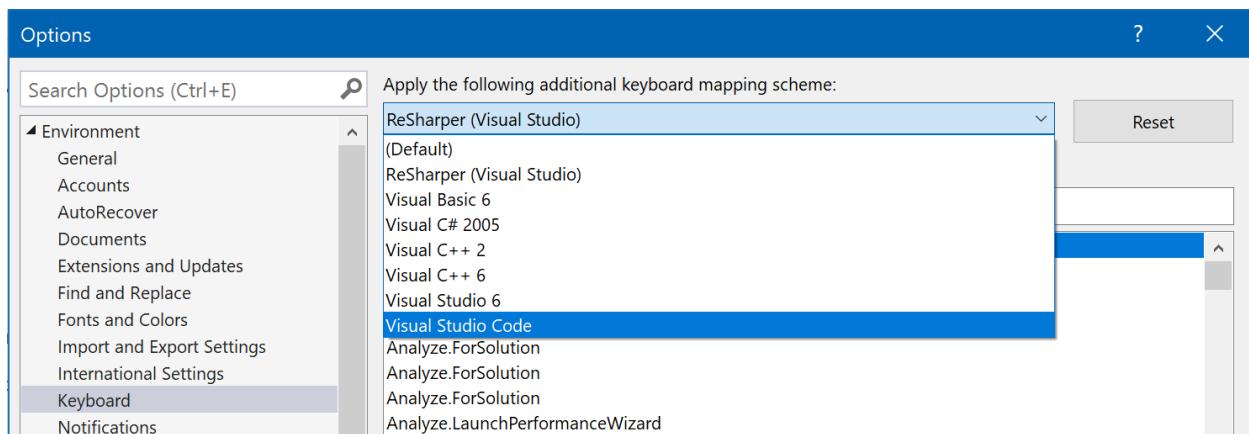
11/26/2019 • 9 minutes to read • [Edit Online](#)

Learn how Visual Studio makes developers more productive than ever. Take advantage of our performance and productivity improvements like navigation to decompiled assemblies, variable name suggestions as you type, a hierarchy-view in **Test Explorer**, Go To All (**Ctrl+T**) to navigate to file/type/member/symbol declarations, an intelligent **Exception Helper**, code style configuration and enforcement, and many refactorings and code fixes.

I'm used to keyboard shortcuts from a different editor

New in Visual Studio 2017 version 15.8

If you're coming from another IDE or coding environment, you can change your keyboard scheme to *Visual Studio Code* or *ReSharper* (*Visual Studio*):



Some extensions also offer keyboard schemes:

- [HotKeys for Visual Studio \(ReSharper/IntelliJ\)](#)
- [Emacs Emulation](#)
- [VSVim](#)

The following are popular Visual Studio shortcuts:

SHORTCUT (ALL PROFILES)	COMMAND	DESCRIPTION
Ctrl+T	Go To All	Navigate to any file, type, member, or symbol declaration
F12 (also Ctrl+Click)	Go To Definition	Navigate to where a symbol is defined
Ctrl+F12	Go To Implementation	Navigate from a base type or member to its various implementations
Shift+F12	Find All References	See all symbol or literal references
Alt+Home	Go To Base	Navigate up the inheritance chain

Shortcut (All Profiles)	Command	Description
Ctrl+. (also Alt+Enter in C# Profile)	Quick Actions and Refactorings	See what code fixes, code generation actions, refactorings, or other quick actions are available at your cursor position or code selection
Ctrl+D	Duplicate line	Duplicates the line of code that the cursor is in (available in Visual Studio 2017 version 15.6 and later)
Shift+Alt++/-	Expand/Contract selection	Expands or contracts the current selection in the editor (available in Visual Studio 2017 version 15.5 and later)
Shift + Alt + .	Insert Next Matching Caret	Adds a selection and caret at the next location that matches the current selection (available in Visual Studio 2017 version 15.8 and later)
Ctrl+Q	Search	Search all Visual Studio settings
F5	Start Debugging	Start debugging your application
Ctrl+F5	Run without Debug	Run your application locally without debugging
Ctrl+K,D (Default Profile) or Ctrl+E,D (C# Profile)	Format Document	Cleans up formatting violations in your file based on your newline, spacing, and indentation settings
Ctrl+\,Ctrl+E (Default Profile) or Ctrl+W,E (C# Profile)	View Error List	See all errors in your document, project, or solution
Alt + PgUp/PgDn	Go to Next/Previous Issue	Jump to the previous/next error, warning, suggestion in your document (available in Visual Studio 2017 version 15.8 and later)
Ctrl+K,/	Toggle single line comment/uncomment	This command adds or removes a single line comment depending on whether your selection is already commented
Ctrl+Shift+/	Toggle block comment/uncomment	This command adds or removes block comments depending on what you have selected

NOTE

Some extensions unbind the default Visual Studio keybindings. To use the above commands, restore your keybindings to Visual Studio's defaults by going to **Tools > Import and Export Settings > Reset all settings** or **Tools > Options > Keyboard > Reset**.

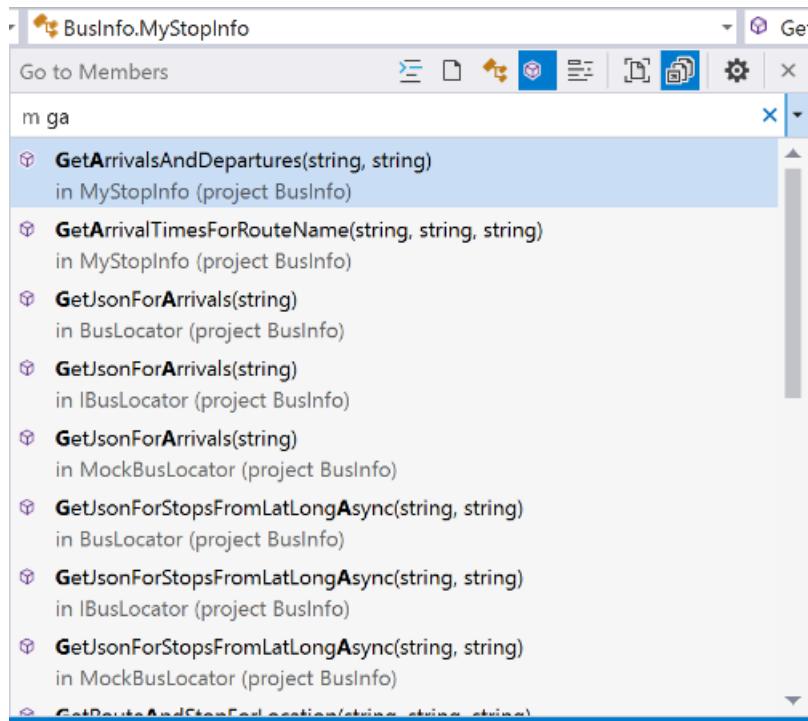
For more information about keyboard shortcuts and commands, see [Productivity shortcuts](#) and [Popular](#)

keyboard shortcuts.

Navigate quickly to files or types

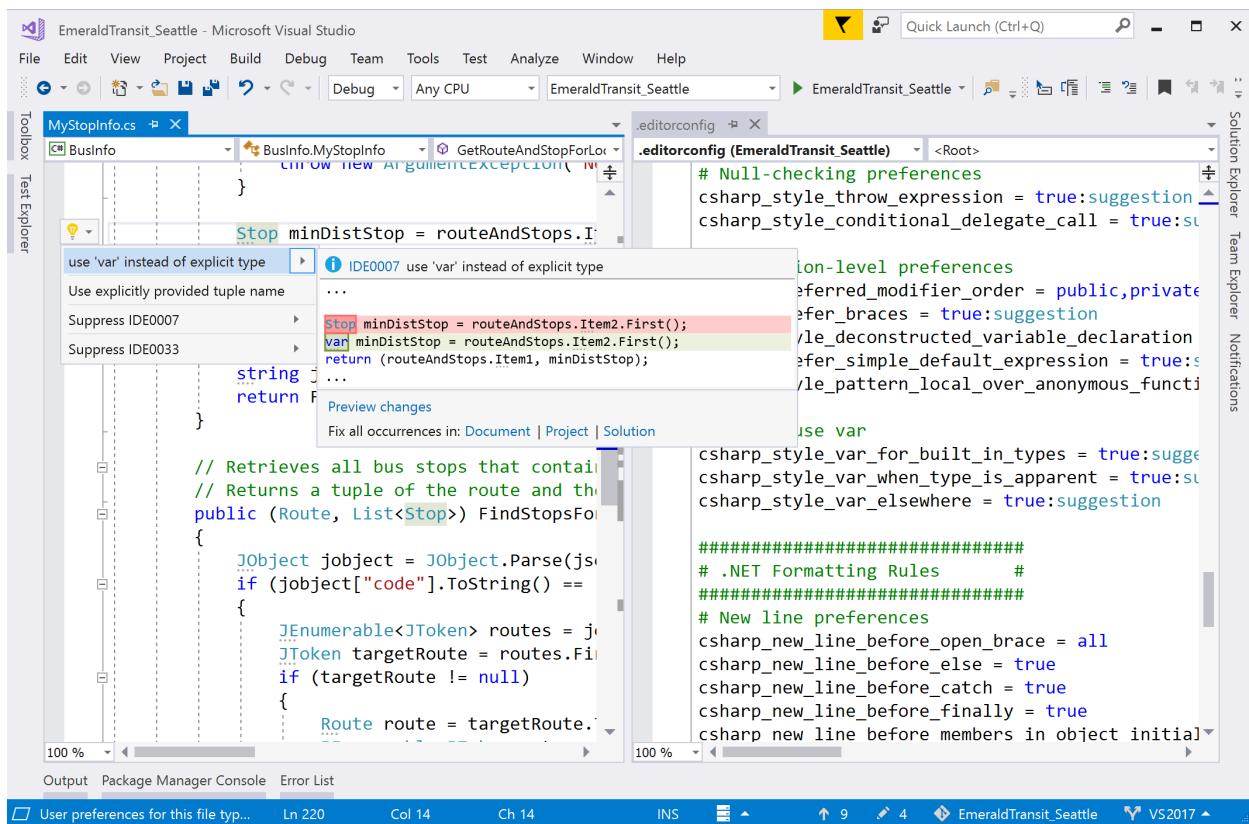
Visual Studio has a feature called **Go To All (Ctrl+T)**. **Go To All** enables you to quickly jump to any file, type, member, or symbol declaration.

- Change the location of this search bar or turn off the live navigation preview by using the **gear** icon.
- Filter results using syntax such as `t mytype`.
- Scope your search to just the current document.
- Camel case matching is supported.

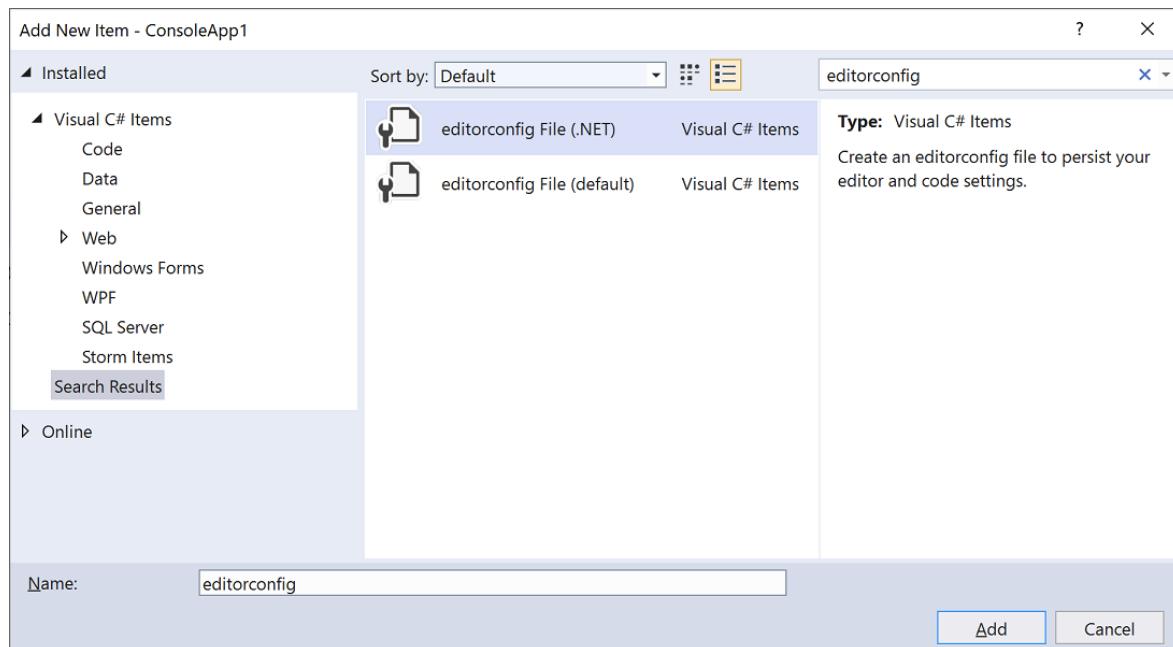


Enforce code style rules

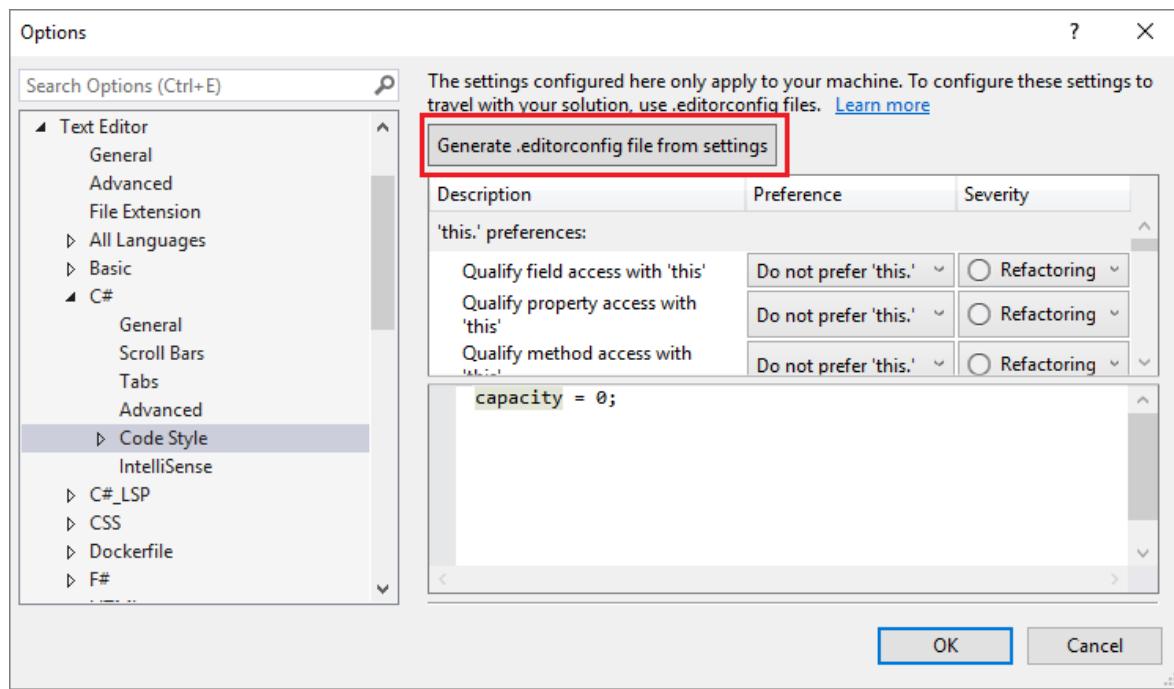
You can use an EditorConfig file to codify coding conventions and have them travel with your source.



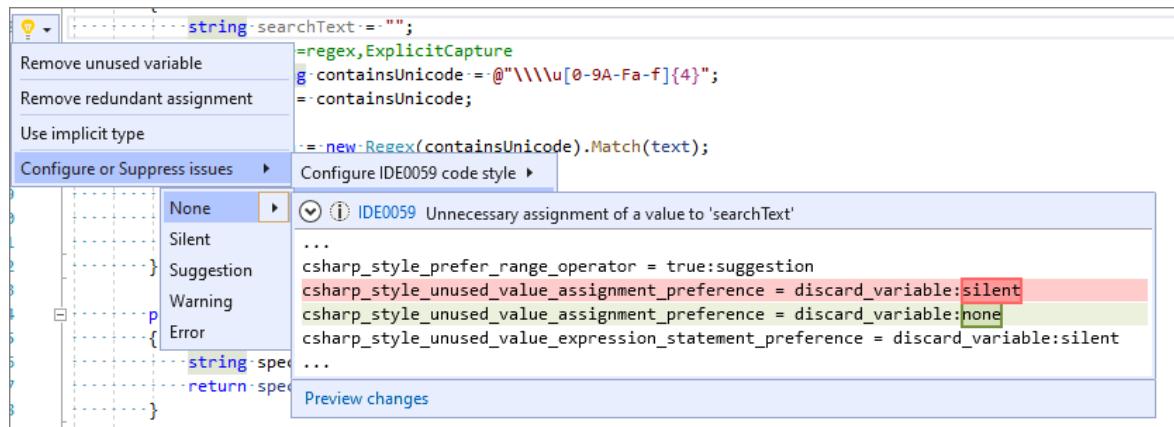
- Add a default or .NET-style EditorConfig file to your project by choosing **Add > New Item**. In the **Add New Item** dialog box, search for "editorconfig". Select either of the **editorconfig File** item templates and then choose **Add**.



- Automatically create an *.editorconfig* file based on your code style settings in **Tools > Options > Text Editor > C# > Code Style**.



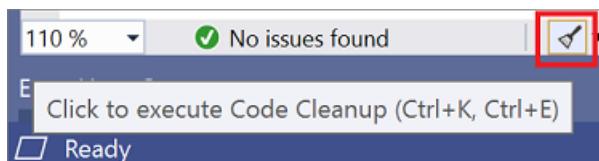
- The [code inference feature](#) of IntelliCode for Visual Studio infers your code styles from existing code. It then creates a non-empty EditorConfig file with your code-style preferences already defined.
- Configure the severity level of a code style rule directly through the editor. If you currently do not have an .editorconfig file, one will be generated for you. Place your cursor on the error, warning, or suggestion and type **Ctrl+.** to open the Quick Actions and Refactorings menu. Select **Configure or Suppress issues**. Then select the rule and choose the severity level you would like to configure for that rule. This will update your existing EditorConfig with the rule's new severity.



Check out the [.NET coding convention options](#) documentation, which also contains an example of a complete EditorConfig file.

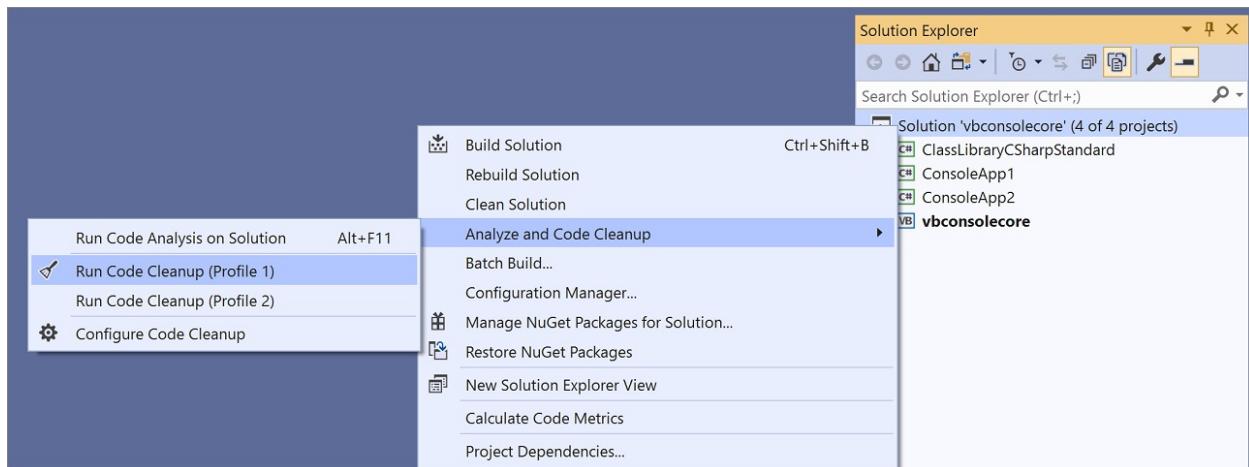
Code Cleanup

Visual Studio provides on-demand formatting of your code file, including code style preferences, through the **Code Cleanup** feature. To run Code Cleanup, click the broom icon at the bottom of the editor or press **Ctrl+K, Ctrl+E**.



You can also run code cleanup across your entire project or solution. Right-click on the project or solution name

in **Solution Explorer**, select **Analyze and Code Cleanup**, and then select **Run Code Cleanup**.



In addition to formatting your file for spaces, indents, et cetera, **Code Cleanup** also applies selected code styles. Your preferences for each code style are read from the [EditorConfig file](#), if you have one for the project, or from the [code style settings](#) in the **Options** dialog box.

Refactorings and code fixes

Visual Studio comes with numerous refactorings, code generation actions, and code fixes. Red squiggles represent errors, green squiggles represent warnings, and three gray dots represent code suggestions. You can access code fixes by clicking the light bulb or screwdriver icon, or by pressing **Ctrl+.** or **Alt+Enter**. Each fix comes with a preview window that shows a live code diff of how the fix works.

Popular quick fixes and refactorings include:

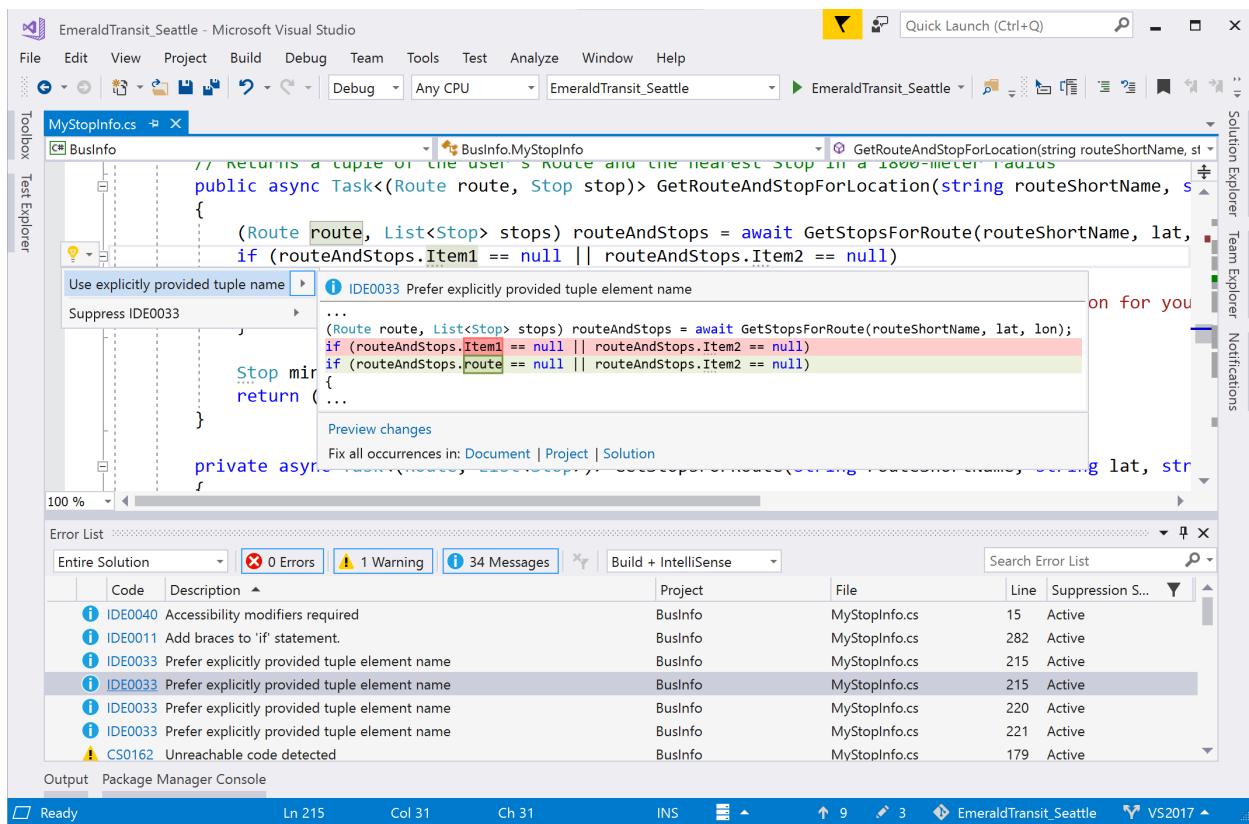
- Rename
- Extract Method
- Change Method Signature
- Generate Constructor
- Generate Method
- Move Type to File
- Add Null-Check
- Add Parameter
- Remove Unnecessary Usings
- Foreach Loop to LINQ Query or to LINQ method
- Pull Members Up

For more information, see [code generation features](#).

You can [install FxCop analyzers](#) to flag code issues. Or, write your own refactoring or code fix with [Roslyn analyzers](#).

Several community members have written free extensions that add additional code inspections:

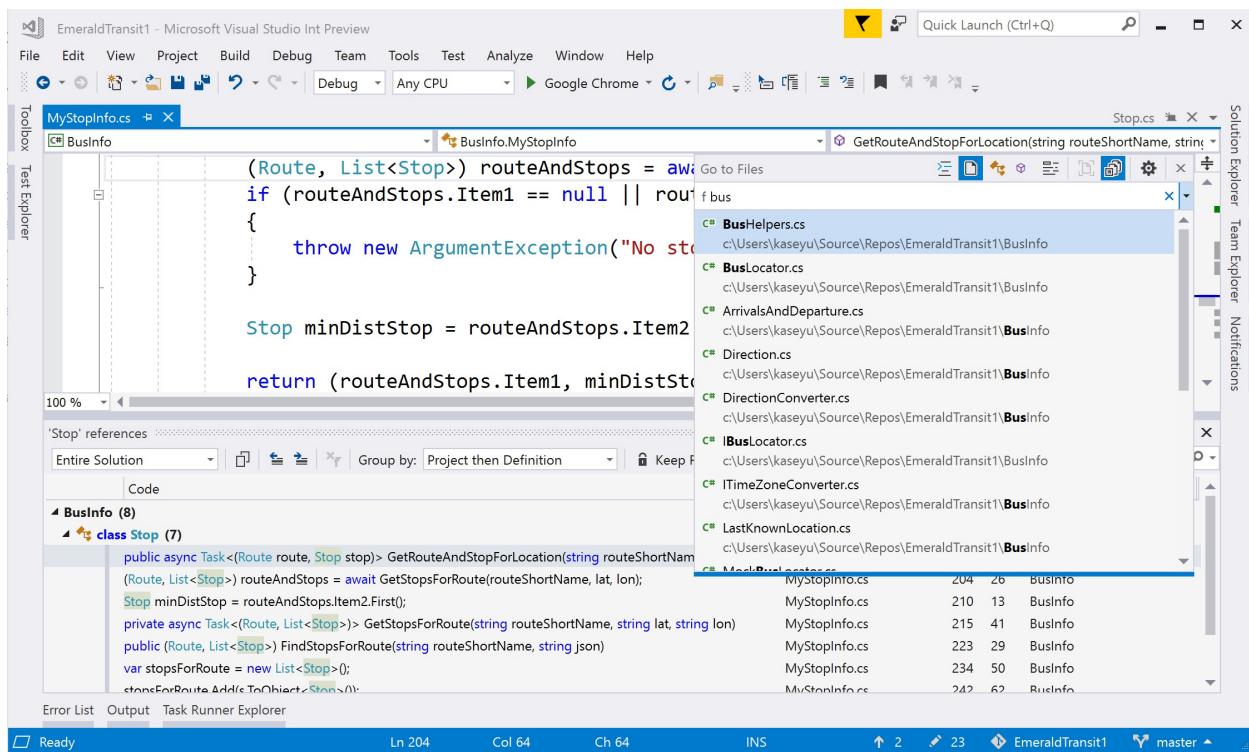
- [Roslynator](#)
- [SonarLint for Visual Studio](#)
- [StyleCopAnalyzers](#)
- [CodeCracker](#)



Find Usages, Go To Implementation, and Navigate To Decompiled Assemblies

Visual Studio has many features to help you search and [navigate your code](#).

FEATURE	SHORTCUT	DETAILS/IMPROVEMENTS
Find All References	Shift+F12	Results are colorized and can be grouped by project, definition, and reference type, such as read or write. You can also "lock" results.
Go To Implementation	Ctrl+F12	You can use Go To Definition on the <code>override</code> keyword to navigate to the overridden member
Go To Definition	F12 or Ctrl+Click	Press Ctrl while clicking to navigate to definition
Peek Definition	Alt+F12	Inline view of a definition
Structure Visualizer	Gray, dotted-lines between braces	Hover to see your code structure
Navigation to decompiled assemblies	F12 or Ctrl+Click	Navigate to external source (decompiled with ILSpy) by enabling the feature: Tools > Options > Text Editor > C# > Advanced > Enable navigation to decompiled sources .



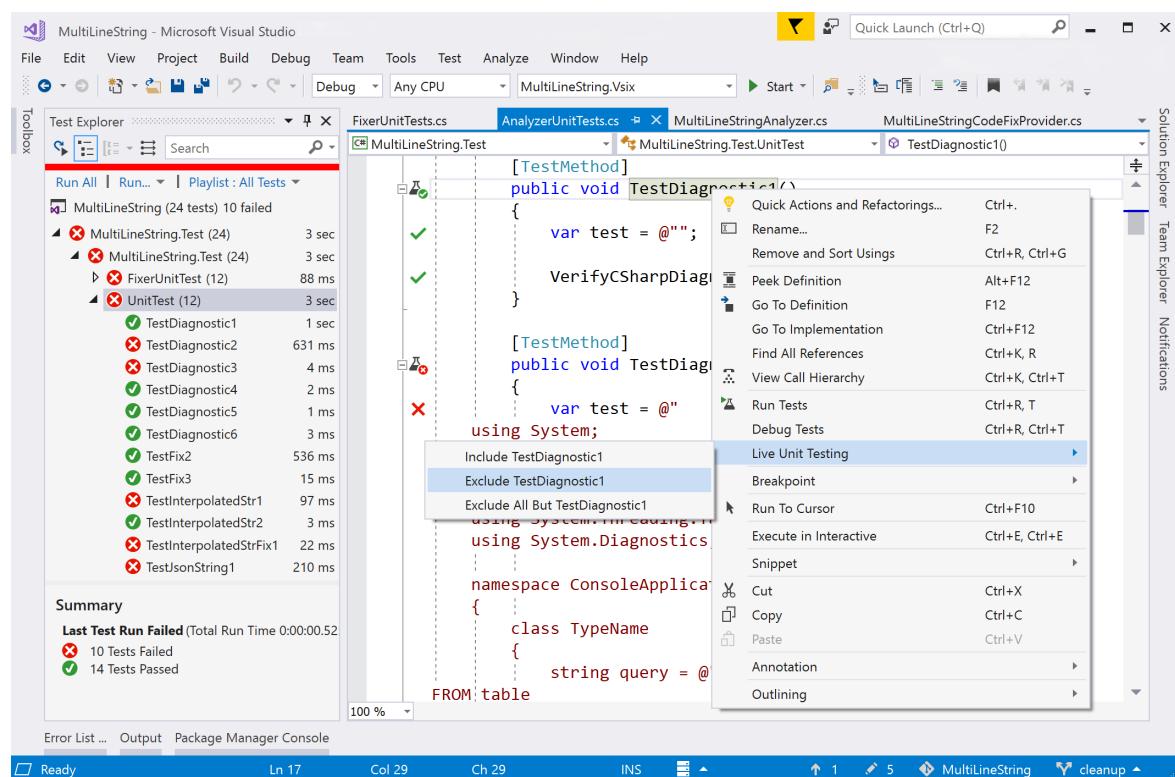
Improved IntelliSense

Use IntelliCode for Visual Studio to get [context-aware code completions](#) instead of just an alphabetical list. You can also train a [custom IntelliSense model](#) based on your own domain-specific libraries.

Unit testing

Starting in Visual Studio 2017, there are numerous improvements to the testing experience. You can test with the MSTest v1, MSTest v2, NUnit, or XUnit test frameworks.

- **Test Explorer** test discovery is fast.
- Organize your tests in **Test Explorer** with *hierarchical sorting*.

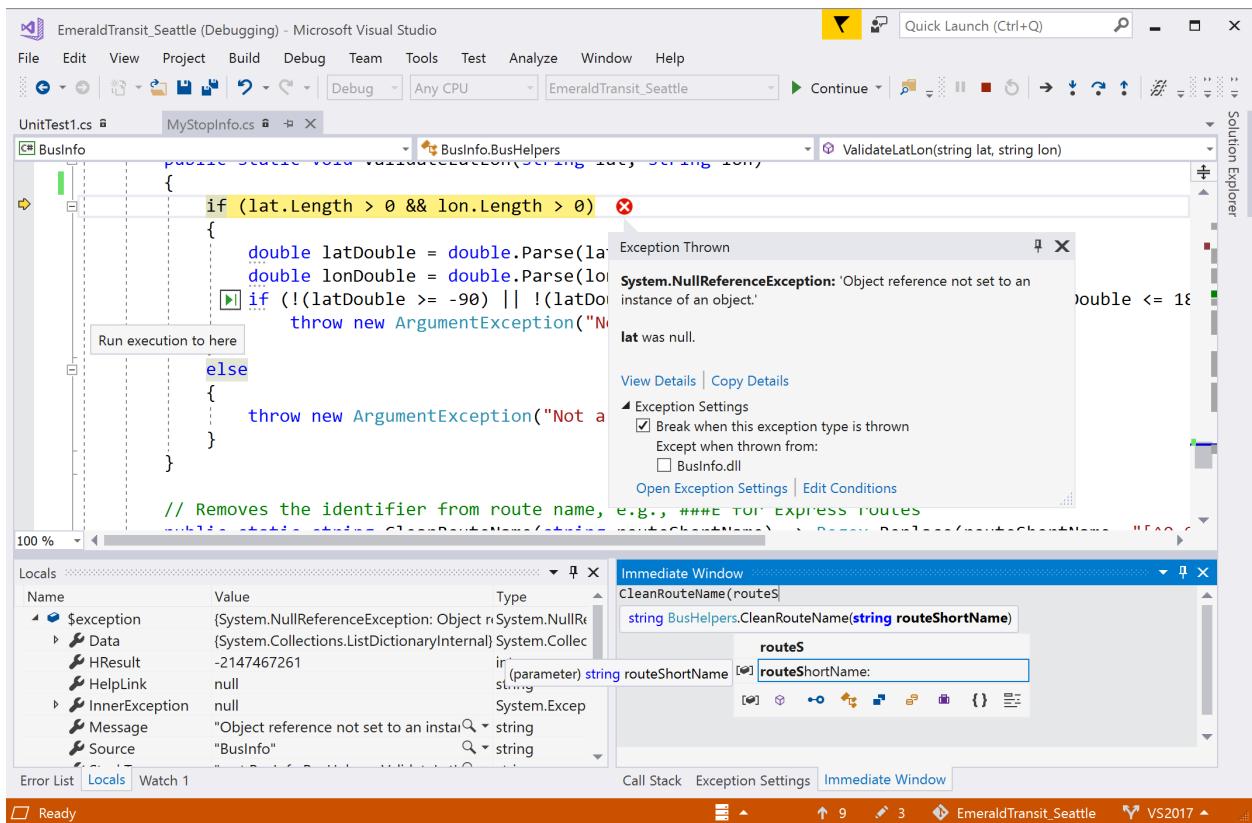


- [Live unit testing](#) continuously runs tests impacted by your code changes and updates inline editor icons to let you know the status of your tests. Include or exclude specific tests or test projects from your live test set. (Visual Studio Enterprise edition only.)

Debugging

Some of Visual Studio's debugging capabilities include:

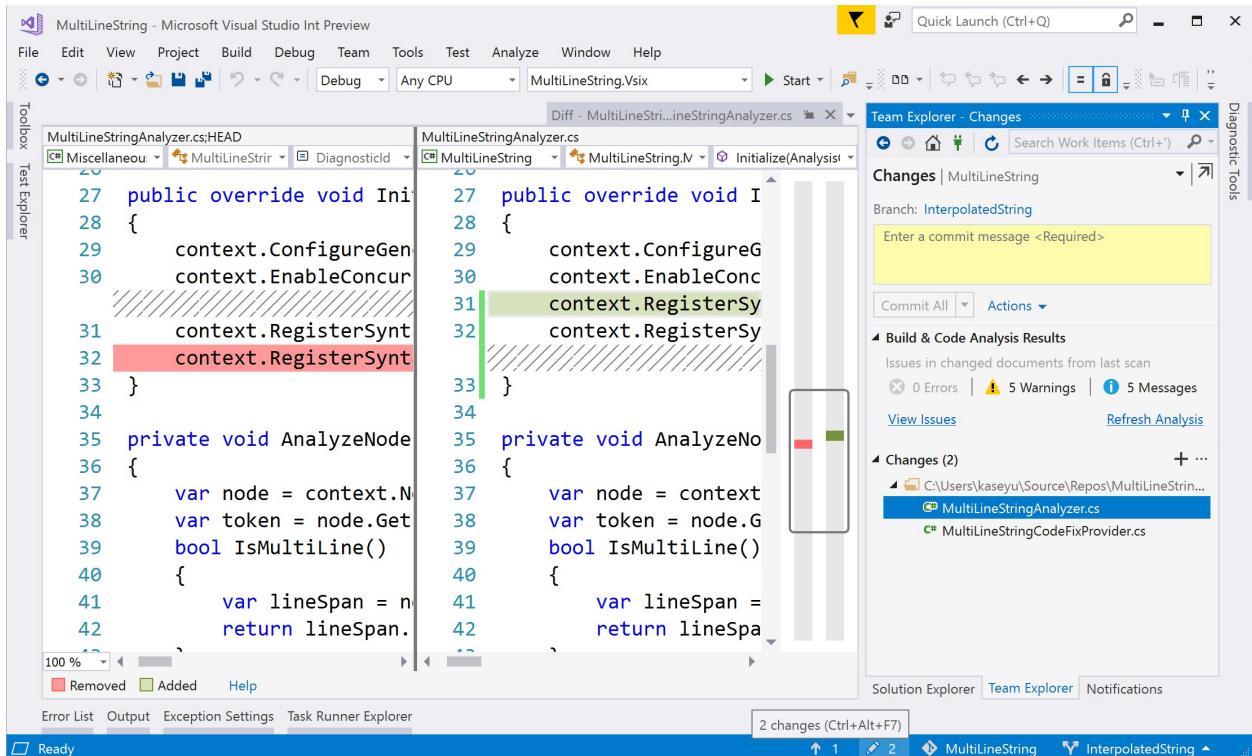
- The ability to search for a string within the **Watch**, **Autos**, and **Locals** windows.
- *Run to click*, which lets you hover next to a line of code, hit the green 'play' icon that appears, and run your program until it reaches that line.
- The **Exception Helper**, which puts the most important information at the top level in the dialog, for example, which variable is `null` in a `NullReferenceException`.
- [Step back debugging](#), which lets you go back to previous breakpoints or steps and view the state of the application as it was in the past.
- [Snapshot debugging](#), which lets you investigate the state of a live web application at the moment an exception was thrown (must be on Azure).
- *Run to click*, which lets you hover next to a line of code, hit the green 'play' icon that appears, and run your program until it reaches that line.
- The **Exception Helper**, which puts the most important information at the top level in the dialog, for example, which variable is `null` in a `NullReferenceException`.
- [Step back debugging](#), which lets you go back to previous breakpoints or steps and view the state of the application as it was in the past.
- [Snapshot debugging](#), which lets you investigate the state of a live web application at the moment an exception was thrown (must be on Azure).



Version control

You can use git or TFVC to store and update your code in Visual Studio.

- Install the [Pull requests for Visual Studio](#) to create, review, check out, and run pull requests without leaving Visual Studio.
- Organize your local changes in [Team Explorer](#) and use the status bar to track pending commits and changes.
- Set up continuous integration and delivery for your ASP.NET projects inside of Visual Studio with the [Continuous delivery tools for Visual Studio](#) extension.



What other features should I know about?

Here is a list of editor and productivity features to make writing code more efficient. Some features may need to be enabled because they are off by default (they may index things on your machine, are controversial, or are currently experimental).

FEATURE	DETAILS	HOW TO ENABLE
Locate File in Solution Explorer	Highlights the active file in Solution Explorer	Tools > Options > Projects and Solutions > Track Active Item in Solution Explorer
Add usings for types in reference assemblies and NuGet packages	Shows an error light bulb with a code fix to install a NuGet package for an unreferenced type	Tools > Options > Text Editor > C# > Advanced > Suggest usings for types in reference assemblies and Suggest usings for types in NuGet packages
Enable full solution analysis	See all errors in your solution in the Error List	Tools > Options > Text Editor > C# > Advanced > Enable full solution analysis
Enable navigation to decompiled sources	Allow Go To Definition on types/members from external sources and use the ILSpy decompiler to show method bodies	Tools > Options > Text Editor > C# > Advanced > Enable navigation to decompiled sources

FEATURE	DETAILS	HOW TO ENABLE
Completion/Suggestion Mode	Changes the completion behavior in IntelliSense. Developers with IntelliJ backgrounds tend to use a non-default setting here.	Menu > Edit > IntelliSense > Toggle Completion Mode
CodeLens	Displays code reference information and change history in the editor. (Source control CodeLens indicators aren't available in Visual Studio Community edition.)	Tools > Options > Text Editor > All Languages > CodeLens
Code snippets	Help stub out common boilerplate code	Type a snippet name and press Tab twice.

Identify and customize keyboard shortcuts in Visual Studio

10/18/2019 • 4 minutes to read • [Edit Online](#)

You can identify keyboard shortcuts for Visual Studio commands, customize those shortcuts, and export them for others to use. Many shortcuts always invoke the same commands, but the behavior of a shortcut can vary based on the following conditions:

- Which default environment settings you choose the first time that you open Visual Studio—for example, General Development or Visual C#. (For information about changing or resetting your settings, see [Environment settings](#).)
- Whether you've customized the shortcut's behavior.
- Which context you're in when you choose the shortcut. For example, the **F2** shortcut invokes the `Edit.EditCell` command if you're using the **Settings Designer** and it invokes the `File.Rename` command if you're using **Team Explorer**.

Regardless of settings, customization, and context, you can always find and change a keyboard shortcut in the **Options** dialog box. You can also look up the default keyboard shortcuts for several dozen commands in [Popular keyboard shortcuts](#). For a complete list of all default shortcuts (based on the **General Development** settings), see [All keyboard shortcuts](#).

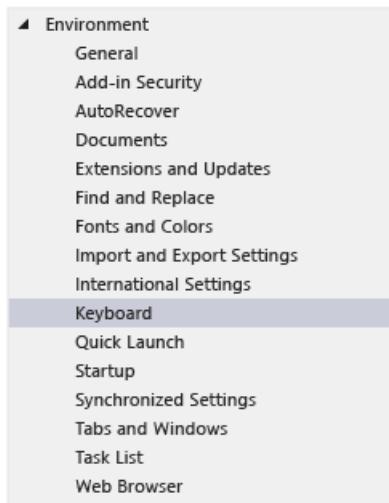
If a shortcut is assigned to a command in the *Global* context and no other contexts, that shortcut will always invoke that command. But a shortcut can be assigned to one command in the Global context and a different command in a specific context. If you use such a shortcut when you're in the specific context, the shortcut invokes the command for the specific context, not the Global context.

NOTE

Your settings and edition of Visual Studio might change the names and locations of menu commands and the options that appear in dialog boxes. This page is based on the **General Development** settings profile.

Identify a keyboard shortcut

1. On the menu bar, choose **Tools** > **Options**.
2. Expand **Environment**, and then choose **Keyboard**.



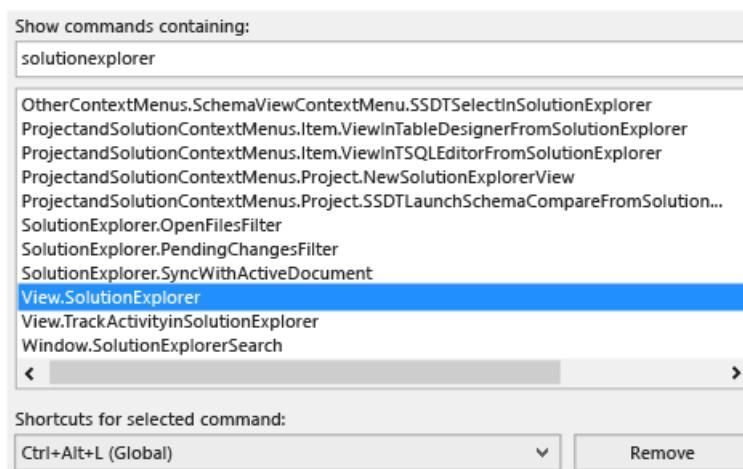
3. In the **Show commands containing** box, enter all or part of the name of the command without spaces.

For example, you can find commands for `solutionexplorer`.

4. In the list, choose the correct command.

For example, you can choose `View.SolutionExplorer`.

5. If the command has a keyboard shortcut, it appears in the **Shortcut(s) for selected command** list.



Customize a keyboard shortcut

1. On the menu bar, choose **Tools > Options**.
2. Expand **Environment**, and then choose **Keyboard**.
3. Optional: Filter the list of commands by entering all or part of the name of the command, without spaces, in the **Show commands containing** box.
4. In the list, choose the command to which you want to assign a keyboard shortcut.

In the **Use new shortcut in** list, choose the feature area in which you want to use the shortcut.

For example, you can choose **Global** if you want the shortcut to work in all contexts. You can use any shortcut that isn't mapped (as Global) in another editor. Otherwise, the editor overrides the shortcut.

NOTE

You can't assign the following keys as part of a keyboard shortcut in **Global**:

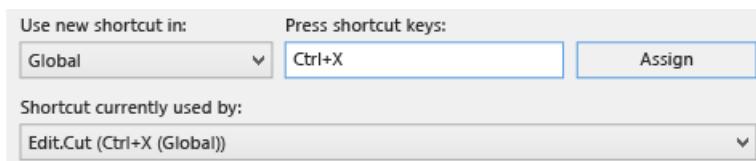
- Enter, Tab, Caps Lock
- Print Scrn/Sys Rq, Scroll Lock, Pause/Break
- Insert, Home, End, Page Up, Page Down
- The Windows logo key, the Application key, any of the Arrow keys
- Num Lock, Delete, or Clear on the numeric keypad
- The Ctrl+Alt+Delete key combination

5. In the **Press shortcut key(s)** box, enter the shortcut that you want to use.

NOTE

You can create a shortcut that combines a letter with the **Alt** key, the **Ctrl** key, or both. You can also create a shortcut that combines the **Shift** key and a letter with the **Alt** key, the **Ctrl** key, or both.

If a shortcut is already assigned to another command, it appears in the **Shortcut currently used by** box. In that case, choose the **Backspace** key to delete that shortcut before you try a different one.



6. Choose the **Assign** button.

NOTE

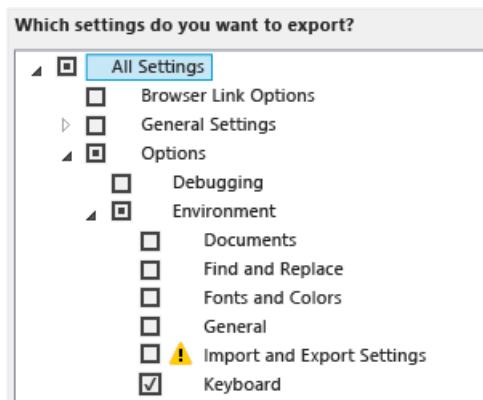
If you specify a different shortcut for a command, click **Assign**, and then click **Cancel** to close the dialog box, the shortcut you assigned is not reverted.

Share custom keyboard shortcuts

You can share your custom keyboard shortcuts by exporting them to a file and then giving the file to others so that they can import the data.

To export only keyboard shortcuts

1. On the menu bar, choose **Tools > Import and Export Settings**.
2. Choose **Export selected environment settings**, and then choose **Next**.
3. Under **What settings do you want to export?**, clear the **All Settings** check box, expand **Options**, and then expand **Environment**.
4. Select the **Keyboard** check box, and then choose **Next**.



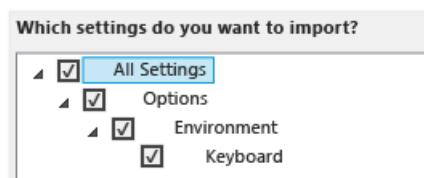
5. In the **What do you want to name your settings file** and **Store my settings file in this directory** boxes, either leave the default values or specify different values, and then choose **Finish**.

By default, your shortcuts are saved in a file in the `%USERPROFILE%\Documents\Visual Studio 2017\Settings` folder. The name of the file reflects the date when you exported the settings, and the extension is `.vssettings`.

By default, your shortcuts are saved in a file in the `%USERPROFILE%\Documents\Visual Studio 2019\Settings` folder. The name of the file reflects the date when you exported the settings, and the extension is `.vssettings`.

To import only keyboard shortcuts

1. On the menu bar, choose **Tools > Import and Export Settings**.
2. Choose the **Import selected environment settings** option button, and then choose **Next**.
3. Choose the **No, just import new settings, overwriting my current settings** option button, and then choose **Next**.
4. Under **My Settings**, choose the file that contains the shortcuts that you want to import, or choose the **Browse** button to locate the correct file.
5. Choose **Next**.
6. Under **Which settings do you want to import?**, clear the **All Settings** check box, expand **Options**, and then expand **Environment**.
7. Select the **Keyboard** check box, and then choose **Finish**.



See also

- [Accessibility features of Visual Studio](#)

How to use the keyboard exclusively

10/18/2019 • 2 minutes to read • [Edit Online](#)

Keyboard shortcuts can make it easier to navigate the Visual Studio IDE and to write code. This article explores a few ways you can use keyboard shortcuts more effectively.

For a full listing of command shortcut keys in Visual Studio, see [Default keyboard shortcuts](#).

TIP

To learn more about accessibility updates, see the [Accessibility improvements in Visual Studio 2017](#) blog post.

NOTE

Depending on your settings or the edition of Visual Studio you use, the dialog boxes and menu commands you see might differ from those described in Help. To change your settings, choose **Import and Export Settings** on the **Tools** menu. For more information, see [Reset settings](#).

Toolbox controls

To add a control on the Toolbox to a form or designer without using the mouse:

1. On the menu bar, choose **View > Toolbox**.
2. Use the **Ctrl+Up arrow** or **Ctrl+Down arrow** keys to move among the sections in the **Toolbox** tab.
3. Use the **Up arrow** key or **Down arrow** key to move among the controls in a section.
4. After you select the control, use the **Enter** key to add the control to the form or designer.

Dialog box options

To move among the options in a dialog box and change option settings by using only the keyboard:

1. Use **Tab** or **Shift+Tab** to move up and down through the controls in the dialog box.
2. To change option settings:
 - For radio buttons, use the **Up arrow** and **Down arrow** keys to change the selection.
 - For check boxes, press **Spacebar** to select or unselect.
 - For drop-down lists, use **Alt+Down arrow** to display items, and then use the **Up arrow** and **Down arrow** keys to change the selected item.
 - For buttons, select **Enter** to invoke.
 - For grids, use the arrow keys to navigate. For drop-down lists in grids, use **Shift+Alt+Down arrow** to display items, and then use the **Up arrow** and **Down arrow** keys to change the selected item.

Navigate between windows and files

- To move among files in an editor or designer, choose the **Ctrl+Tab** keyboard shortcut to display the IDE

Navigator with **Active Files** selected. Choose the **Enter** key to navigate to the highlighted file.

- To move among docked tool windows, choose the **Alt+F7** keyboard shortcut to display the IDE Navigator with **Active Tool Windows** selected. Choose the **Enter** key to navigate to the highlighted window.

Move and dock tool windows

1. Navigate to the tool window you intend to move and give it focus.

2. On the **Window** menu, select the **Dockable** option.

3. Press **Alt+Spacebar**, and then choose **Move**.

The docking guide diamond appears.

4. Use the arrow keys to move the window to a new location.

The mouse pointer moves with the window as you use the arrow keys.

5. When you've reached the new location, use the arrow keys to move the mouse pointer over the correct portion of the guide diamond.

An outline of the tool window appears in the new docking location.

6. Press **Enter**.

The tool window snaps into place at the new docking location.

See also

- [Identifying and customizing keyboard shortcuts](#)
- [Accessibility tips and tricks](#)
- [Default keyboard shortcuts](#)
- [Accessibility in Microsoft products](#)

Windows Forms Designer overview

10/18/2019 • 2 minutes to read • [Edit Online](#)

Windows Forms Designer in Visual Studio provides a rapid development solution for creating Windows Forms-based applications. Windows Forms Designer lets you easily add controls to a form, arrange them, and write code for their events. For more information about Windows Forms, see [Windows Forms overview](#).

Functionality

Using the designer you can:

- Add components, data controls, or Windows-based controls to a form.
- Double-click the form in the designer and write code in the `Load` event for that form, or double-click a control on the form and write code for the control's default event.
- Edit a control's Text property by selecting the control and typing a name.
- Adjust the placement of the selected control by moving it with the mouse or the arrow keys. Similarly, adjust the placement more precisely using the Ctrl and arrow keys. Finally, adjust the size of the control by using the Shift and arrow keys.
- Select multiple controls by selecting either **Shift** or **Ctrl** while you click. When using sing **Shift**+click, the first control selected is the dominant control when aligning or manipulating size. When using **Ctrl**+click, the last control selected is dominant, so the dominant control changes with every new control added. Alternatively, you can select multiple controls by dragging a selection rectangle around the controls that you want to select.

NOTE

Use Windows Forms Designer, and not the Resource Editor, to make changes to a form's resource (.resx) file. If you edit a form-based .resx file, you'll see a warning that changes you make in the Resource Editor may be lost. This is because the Windows Forms Designer generates the .resx file.

See also

- [Windows Forms overview](#)
- [Windows Forms controls](#)
- [User input in Windows Forms](#)
- [Data binding in Windows Forms](#)
- [Enhance Windows Forms apps](#)
- [System.Windows.Forms API reference](#)

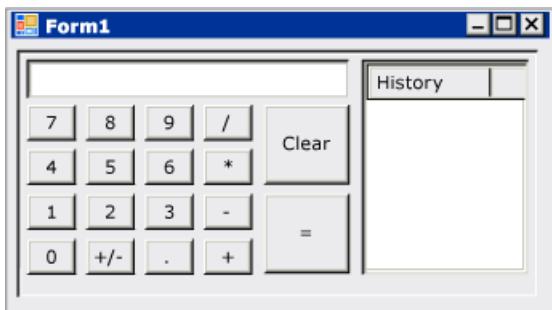
Walkthrough: Get started with Windows Forms Designer

10/18/2019 • 10 minutes to read • [Edit Online](#)

The Windows Forms Designer provides many tools for building Windows Forms applications. This article illustrates how to build an app using the various tools provided by the designer, including the following tasks:

- Arrange controls using snaplines.
- Accomplish designer tasks using smart tags.
- Set margins and padding for controls.
- Arrange controls using a `TableLayoutPanel` control.
- Partition your control's layout by using a `SplitContainer` control.
- Navigate your layout with the Document Outline window.
- Position controls with the size and location information display.
- Set property values using the Properties window.

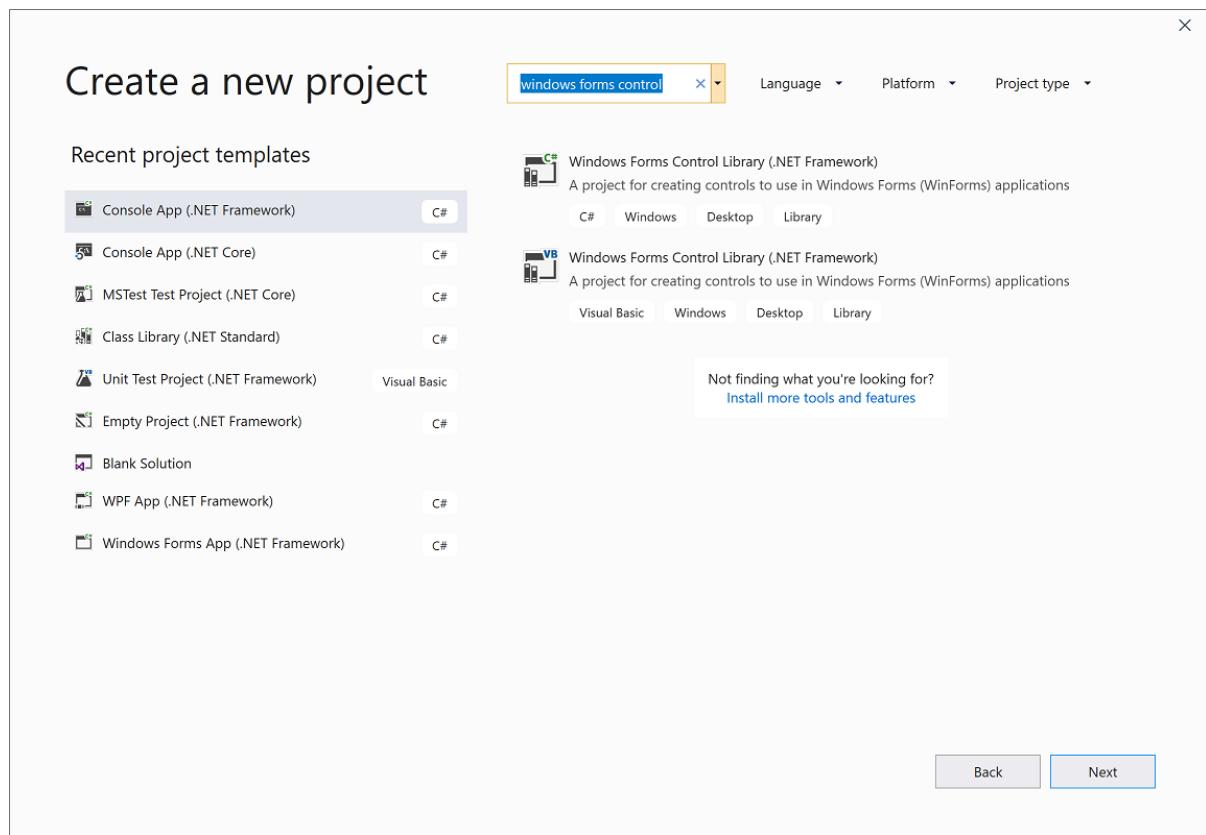
When you're finished, you'll have a custom control that's been assembled using many of the layout features available in the Windows Forms Designer. This control implements the user interface (UI) for a simple calculator. The following image shows the general layout of the calculator control:



Create the custom control project

The first step is to create the DemoCalculator control project.

1. Open Visual Studio and create a new **Windows Forms Control Library** project. Name the project **DemoCalculatorLib**.



2. To rename the file, in **Solution Explorer**, right-select **UserControl1.vb** or **UserControl1.cs**, select **Rename**, and change the file name to DemoCalculator.vb or DemoCalculator.cs. Select **Yes** when you are asked if you want to rename all references to the code element "UserControl1".

The Windows Forms Designer shows the designer surface for the DemoCalculator control. In this view, you can graphically design the appearance of the control by selecting controls and components from Toolbox and placing them on the designer surface. For more information about custom controls, see [Varieties of custom controls](#).

Design the control layout

The DemoCalculator control contains several Windows Forms controls. In this procedure, you'll arrange the controls using the Windows Forms Designer.

1. In the Windows Forms Designer, change the DemoCalculator control to a larger size by selecting the sizing handle in the lower-right corner and dragging it down and to the right. In the lower-right corner of Visual Studio, find the size and location information for controls. Set the size of the control to width 500 and height 400 by watching the size information as you resize the control.
2. In **Toolbox**, select the **Containers** node to open it. Select the **SplitContainer** control and drag it onto the designer surface.

The **SplitContainer** is placed on the DemoCalculator control's designer surface.

TIP

The **SplitContainer** control sizes itself to fit the size of the DemoCalculator control. Look at the **Properties** window to see the property settings for the **SplitContainer** control. Find the **Dock** property. Its value is **DockStyle.Fill**, which means the **SplitContainer** control will always size itself to the boundaries of the DemoCalculator control. Resize the DemoCalculator control to verify this behavior.

3. In the **Properties** window, change the value of the **Dock** property to **None**.

The **SplitContainer** control shrinks to its default size and no longer follows the size of the DemoCalculator

control.

4. Select the smart tag glyph (□) on the upper-right corner of the `SplitContainer` control. Select **Dock in Parent Container** to set the `Dock` property to `Fill`.

The `SplitContainer` control docks to the `DemoCalculator` control's boundaries.

NOTE

Several controls offer smart tags to facilitate design. For more information, see [Walkthrough: Perform common tasks using Smart Tags on Windows Forms controls](#).

5. Select the vertical border between the panels and drag it to the right, so that most of the space is taken by the left panel.

The `SplitContainer` divides the `DemoCalculator` control into two panels with a movable border separating them. The panel on the left will hold the calculator buttons and display, and the panel on the right will show a record of the arithmetic operations performed by the user.

6. In the **Properties** window, change the value of the `BorderStyle` property to `Fixed3D`.
7. In **Toolbox**, select the **Common Controls** node to open it. Select the `ListView` control and drag it into the right panel of the `SplitContainer` control.
8. Select the `ListView` control's smart tag glyph. In the smart tag panel, change the `View` setting to `Details`.
9. In the smart tag panel, select **Edit Columns**.

The **ColumnHeader Collection Editor** dialog box opens.

10. In the **ColumnHeader Collection Editor** dialog box, select **Add** to add a column to the `ListView` control. Change the value of the column's `Text` property to `History`. Select **OK** to create the column.
11. In the smart tag panel, select **Dock in Parent Container**, and then select the smart tag glyph to close the smart tag panel.
12. From the **Containers** node **Toolbox**, drag a `TableLayoutPanel` control into the left panel of the `SplitContainer` control.

The `TableLayoutPanel` control appears on the designer surface with its smart tag panel open. The `TableLayoutPanel` control arranges its child controls in a grid. The `TableLayoutPanel` control will hold the `DemoCalculator` control's display and buttons. For more information, see [Walkthrough: Arrange controls using a TableLayoutPanel](#).

13. Select **Edit Rows and Columns** on the smart tag panel.

The **Column and Row Styles** dialog box opens.

14. Select the **Add** button until five columns are displayed. Select all five columns, and then select **Percent** in the **Size Type** box. Set the **Percent** value to **20**. This sets each column to the same width.
15. Under **Show**, select **Rows**.
16. Select **Add** until five rows are displayed. Select all five rows, and then select **Percent** in the **Size Type** box. Set the **Percent** value to **20**. This sets each row to the same height.
17. Select **OK** to accept your changes, and then select the smart tag glyph to close the smart tag panel.
18. In the **Properties** window, change the value of the `Dock` property to `Fill`.

Populate the control

Now that the layout of the control is set up, you can populate the DemoCalculator control with buttons and a display.

1. In **Toolbox**, double-click the **TextBox** control icon.

A **TextBox** control is placed in the first cell of the **TableLayoutPanel1** control.

2. In the **Properties** window, change the value of the **TextBox** control's **ColumnSpan** property to **5**.

The **TextBox** control moves to a position that is centered in its row.

3. Change the value of the **TextBox** control's **Anchor** property to **Left, Right**.

The **TextBox** control expands horizontally to span all five columns.

4. Change the value of the **TextBox** control's **TextAlign** property to **Right**.

5. In the **Properties** window, expand the **Font** property node. Set **Size** to **14**, and set **Bold** to **true** for the **TextBox** control.

6. Select the **TableLayoutPanel1** control.

7. In **Toolbox**, double-click the **Button** icon.

A **Button** control is placed in the next open cell of the **TableLayoutPanel1** control.

8. In **Toolbox**, double-click the **Button** icon four more times to populate the second row of the **TableLayoutPanel1** control.

9. Select all five **Button** controls by selecting them while holding down the **Shift** key. Press **Ctrl+C** to copy the **Button** controls to the clipboard.

10. Press **Ctrl+V** three times to paste copies of the **Button** controls into the remaining rows of the **TableLayoutPanel1** control.

11. Select all 20 **Button** controls by selecting them while holding down the **Shift** key.

12. In the **Properties** window, change the value of the **Dock** property to **Fill**.

All the **Button** controls dock to fill their containing cells.

13. In the **Properties** window, expand the **Margin** property node. Set the value of **All** to **5**.

All the **Button** controls are sized smaller to create a larger margin between them.

14. Select **button10** and **button20**, and then press **Delete** to remove them from the layout.

15. Select **button5** and **button15**, and then change the value of their **RowSpan** property to **2**. These will be the **Clear** and = buttons for the DemoCalculator control.

Use the Document Outline window

When your control or form is populated with several controls, you may find it easier to navigate your layout with the Document Outline window.

1. On the menu bar, choose **View > Other Windows > Document Outline**.

The Document Outline window shows a tree view of the DemoCalculator control and its constituent controls. Container controls like the **SplitContainer** show their child controls as subnodes in the tree. You

can also rename controls in place using the Document Outline window.

2. In the **Document Outline** window, right-select **button1**, and then select **Rename**. Change its name to **sevenButton**.
3. Using the **Document Outline** window, rename the **Button** controls from the designer-generated name to the production name according to the following list:
 - button1 to **sevenButton**
 - button2 to **eightButton**
 - button3 to **nineButton**
 - button4 to **divisionButton**
 - button5 to **clearButton**
 - button6 to **fourButton**
 - button7 to **fiveButton**
 - button8 to **sixButton**
 - button9 to **multiplicationButton**
 - button11 to **oneButton**
 - button12 to **twoButton**
 - button13 to **threeButton**
 - button14 to **subtractionButton**
 - button15 to **equalsButton**
 - button16 to **zeroButton**
 - button17 to **changeSignButton**
 - button18 to **decimalButton**
 - button19 to **additionButton**
4. Using the **Document Outline** and **Properties** windows, change the **Text** property value for each **Button** control name according to the following list:
 - Change the sevenButton control text property to **7**
 - Change the eightButton control text property to **8**
 - Change the nineButton control text property to **9**
 - Change the divisionButton control text property to **/** (forward slash)
 - Change the clearButton control text property to **Clear**
 - Change the fourButton control text property to **4**
 - Change the fiveButton control text property to **5**
 - Change the sixButton control text property to **6**
 - Change the multiplicationButton control text property to ***** (asterisk)
 - Change the oneButton control text property to **1**

- Change the twoButton control text property to **2**
 - Change the threeButton control text property to **3**
 - Change the subtractionButton control text property to - (hyphen)
 - Change the equalsButton control text property to = (equals sign)
 - Change the zeroButton control text property to **0**
 - Change the changeSignButton control text property to **+/-**
 - Change the decimalButton control text property to . (period)
 - Change the additionButton control text property to + (plus sign)
5. On the designer surface, select all the `Button` controls by selecting them while holding down the **Shift** key.
6. In the **Properties** window, expand the `Font` property node. Set `Size` to **14**, and set `Bold` to **true** for all the `Button` controls.

This completes the design of the DemoCalculator control. All that remains is to provide the calculator logic.

Implement event handlers

The buttons on the DemoCalculator control have event handlers that can be used to implement much of the calculator logic. The Windows Forms Designer enables you to implement the stubs of all the event handlers for all the buttons with one double-click.

1. On the designer surface, select all the `Button` controls by selecting them while holding down the **Shift** key.
2. Double-click one of the `Button` controls.

The Code Editor opens to the event handlers generated by the designer.

Test the control

Because the DemoCalculator control inherits from the `UserControl` class, you can test its behavior with the **UserControl Test Container**. For more information, see [How to: Test the run-time behavior of a UserControl](#).

1. Press **F5** to build and run the DemoCalculator control in the **UserControl Test Container**.
2. Select the border between the `SplitContainer` panels and drag it left and right. The `TableLayoutPanel` and all its child controls resize themselves to fit in the available space.
3. When you are finished testing the control, select **Close**.

Use the control on a form

The DemoCalculator control can be used in other composite controls or on a form. The following procedure describes how to use it.

Create the project

The first step is to create the application project. You'll use this project to build the application that shows your custom control.

1. Create a new **Windows Forms Application** project and name it **DemoCalculatorTest**.
2. In **Solution Explorer**, right-click the **DemoCalculatorTest** project, and then select **Add Reference** to open the **Add Reference** dialog box.

3. Select the **Projects** tab, and then double-click the DemoCalculatorLib project to add the reference to the test project.
4. In **Solution Explorer**, right-click **DemoCalculatorTest**, and then select **Set as StartUp Project**.
5. In the Windows Forms Designer, increase the size of the form to about **700 x 500**.

Use the control in the form's layout

To use the DemoCalculator control in an application, you need to place it on a form.

1. In **Toolbox**, expand the **DemoCalculatorLib Components** node.
2. Drag the **DemoCalculator** control from **Toolbox** onto your form. Move the control to the upper-left corner of the form. When the control is close to the form's borders, *snaplines* will appear. Snaplines indicate the distance of the form's `Padding` property and the control's `Margin` property. Position the control at the location indicated by the snaplines.

For more information, see [Walkthrough: Arrange controls using snaplines](#).

3. Drag a `Button` control from **Toolbox** and drop it onto the form.
4. Move the `Button` control around the DemoCalculator control and observe where the snaplines appear. You can align your controls precisely and easily using this feature. Delete the `Button` control when you're finished.
5. Right-select the DemoCalculator control, and then select **Properties**.
6. Change the value of the `Dock` property to `Fill`.
7. Select the form, and then expand the `Padding` property node. Change the value of **All** to **20**.

The size of the DemoCalculator control is reduced to accommodate the new `Padding` value of the form.

8. Resize the form by dragging the various sizing handles to different positions. Observe how the DemoCalculator control is resized to fit.

Next steps

This article has demonstrated how to construct the user interface for a simple calculator. To continue, you can extend its functionality by implementing the calculator logic, then [publish the app using ClickOnce](#). Or, continue on to a different tutorial where you [create a picture viewer using Windows Forms](#).

See also

- [Windows Forms controls](#)
- [Accessibility for Windows Forms controls](#)
- [Publish using ClickOnce](#)

Disable DPI-awareness in Visual Studio

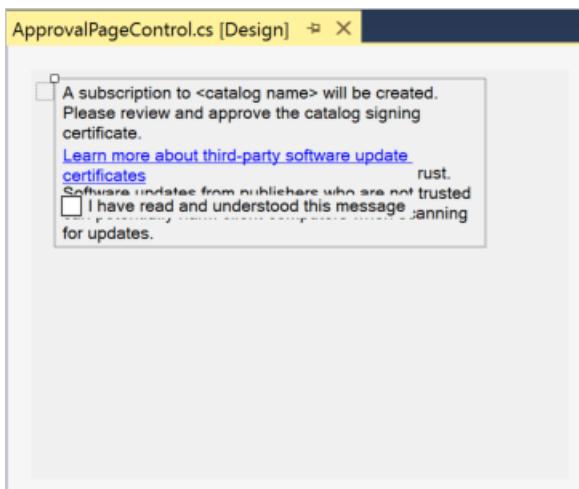
10/18/2019 • 4 minutes to read • [Edit Online](#)

Visual Studio is a dots per inch (DPI) aware application, which means the display scales automatically. If an application states that it's not DPI-aware, the operating system scales the application as a bitmap. This behavior is also called DPI virtualization. The application still thinks that it's running at 100% scaling, or 96 dpi.

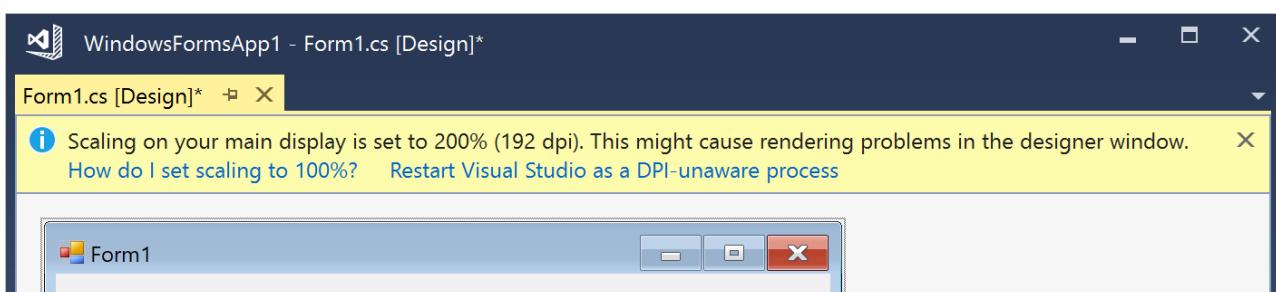
This article discusses the limitations of Windows Forms Designer on HDPI monitors and how to run Visual Studio as a DPI-unaware process.

Windows Forms Designer on HDPI monitors

The **Windows Forms Designer** in Visual Studio doesn't have scaling support. This causes display issues when you open some forms in the **Windows Forms Designer** on high dots per inch (HDPI) monitors. For example, controls can appear to overlap as shown in the following image:



When you open a form in the **Windows Forms Designer** in Visual Studio on an HDPI monitor, Visual Studio displays a yellow informational bar at the top of the designer:



The message reads **Scaling on your main display is set to 200% (192 dpi). This might cause rendering problems in the designer window.**

NOTE

This informational bar was introduced in Visual Studio 2017 version 15.8.

If you aren't working in the designer and don't need to adjust the layout of your form, you can ignore the informational bar and continue working in the code editor or in other types of designers. (You can also [disable notifications](#) so that the informational bar doesn't continue to appear.) Only the **Windows Forms Designer** is

affected. If you do need to work in the **Windows Forms Designer**, the next section helps you [resolve the problem](#).

To resolve the display problem

There are three options to resolve the display problem:

- [Restart Visual Studio as a DPI-unaware process](#)
- [Add a registry entry](#)
- [Set your display scaling setting to 100%](#)

Restart Visual Studio as a DPI-unaware process

You can restart Visual Studio as a DPI-unaware process by selecting the option on the yellow informational bar. This is the preferred way of resolving the problem.

When Visual Studio runs as a DPI-unaware process, the designer layout issues are resolved, but fonts may appear blurry. Visual Studio displays a different yellow informational message when it runs as a DPI-unaware process that says **Visual Studio is running as a DPI-unaware process. WPF and XAML designers might not display correctly.** The informational bar also provides an option to **Restart Visual Studio as a DPI-aware process**.

NOTE

- If you had undocked tool windows in Visual Studio when you selected the option to restart as a DPI-unaware process, the position of those tool windows may change.
- If you use the default Visual Basic profile, or if you have the **Save new projects when created** option deselected in **Tools > Options > Projects and Solutions**, Visual Studio cannot reopen your project when it restarts as a DPI-unaware process. However, you can open the project by selecting it under **File > Recent Projects and Solutions**.

It's important to restart Visual Studio as a DPI-aware process when you're finished working in the **Windows Forms Designer**. When it's running as a DPI-unaware process, fonts can look blurry and you may see issues in other designers such as the **XAML Designer**. If you close and reopen Visual Studio when it's running in DPI-unaware mode, it becomes DPI-aware again. You can also click the **Restart Visual Studio as a DPI-aware process** option in the informational bar.

Add a registry entry

You can mark Visual Studio as DPI-unaware by modifying the registry. Open **Registry Editor** and add an entry to the **HKEY_CURRENT_USER\Software\Microsoft\Windows NT\CurrentVersion\AppCompatFlags\Layers** subkey:

Entry: Depending on whether you're using Visual Studio 2017 or 2019, use one of these values:

- C:\Program Files (x86)\Microsoft Visual Studio\2017\Community\Common7\IDE\devenv.exe
- C:\Program Files (x86)\Microsoft Visual Studio\2019\Community\Common7\IDE\devenv.exe

NOTE

If you're using the Professional or Enterprise edition of Visual Studio, replace **Community** with **Professional** or **Enterprise** in the entry. Also replace the drive letter as necessary.

Type: REG_SZ

Value: DPIUNAWARE

NOTE

Visual Studio remains in DPI-unaware mode until you remove the registry entry.

Set your display scaling setting to 100%

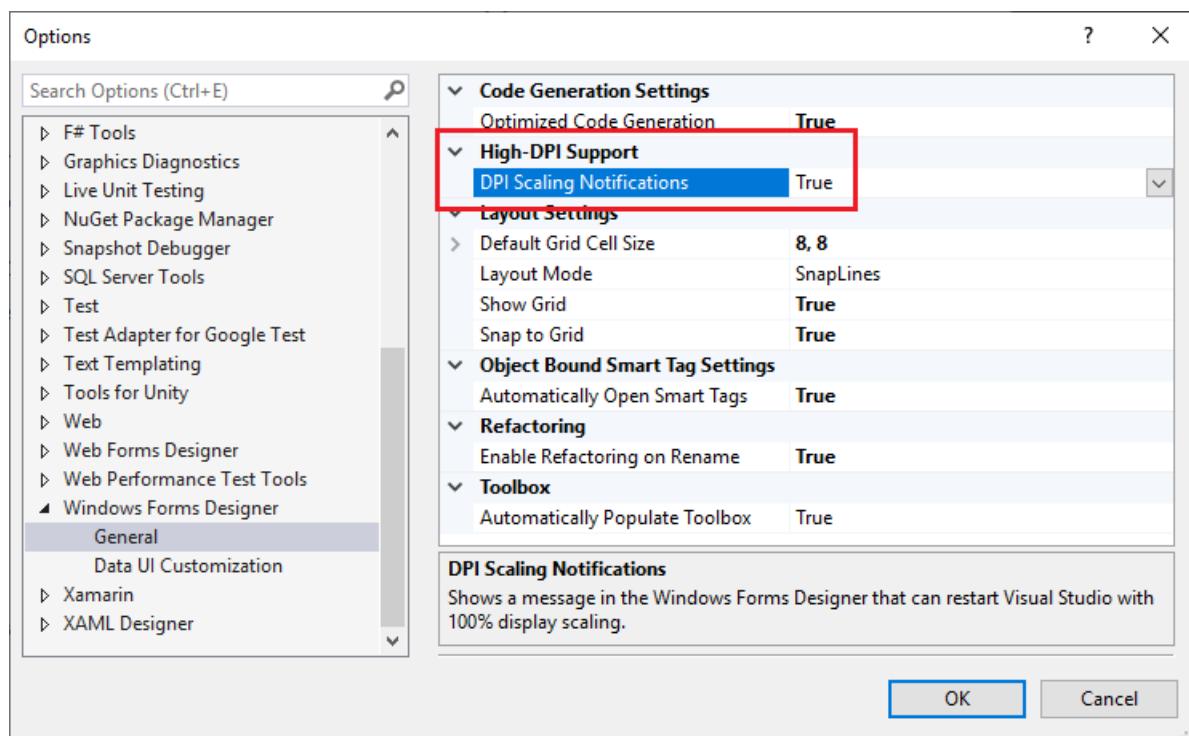
To set your display scaling setting to 100% in Windows 10, type **display settings** in the task bar search box, and then select **Change display settings**. In the **Settings** window, set **Change the size of text, apps, and other items** to **100%**.

Setting your display scaling to 100% may be undesirable, because it can make the user interface too small to be usable.

Disable notifications

You can choose not to be notified of DPI scaling issues in Visual Studio. You might want to disable notifications if you aren't working in the designer, for example.

To disable notifications, choose **Tools > Options** to open the **Options** dialog. Then, choose **Windows Forms Designer > General**, and set **DPI Scaling Notifications** to **False**.



If you want to later reenable scaling notifications, set the property to **True**.

Troubleshoot

If the DPI-awareness transition isn't working as expected in Visual Studio, check to see if you have the `dpiAwareness` value in the **HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Image File Execution Options\devenv.exe** subkey in Registry Editor. Delete the value if it's present.

See also

- [Automatic scaling in Windows Forms](#)

Tutorial 1: Create a picture viewer

10/16/2019 • 2 minutes to read • [Edit Online](#)

In this tutorial, you build an app that loads a picture from a file and displays it in a window. You learn how to use the **Windows Forms Designer** to drag controls like buttons and picture boxes on to your form, set their properties, and use containers to smoothly resize the form. You also get started writing code.

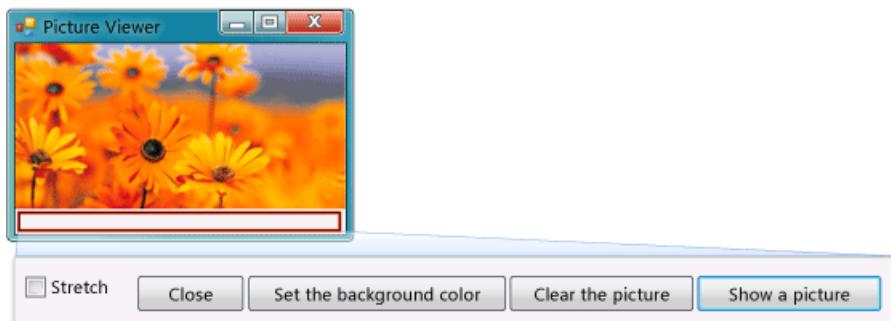
NOTE

This tutorial covers both C# and Visual Basic, so focus on the information that's specific to the programming language you're using.

This tutorial walks you through the following tasks:

- Create a new project.
- Test (debug) an application.
- Add basic controls like check boxes and buttons to a form.
- Position controls on a form by using layouts.
- Add **Open File** and **Color** dialog boxes to a form.
- Write code by using IntelliSense and code snippets.
- Write event handler methods.

When you finish, your app should look similar to the following image:



Tutorial links

TITLE	DESCRIPTION
Step 1: Create a Windows Forms App project	Begin by creating a Windows Forms App project.
Step 2: Run your picture viewer app	Run the Windows Forms App project that you created in the previous step.
Step 3: Set your form properties	Change the way your form looks using the Properties window.

TITLE	DESCRIPTION
Step 4: Lay out your form with a TableLayoutPanel control	Add a <code>TableLayoutPanel</code> control to your form.
Step 5: Add controls to your form	Add controls, such as a <code>PictureBox</code> control and a <code>CheckBox</code> control, to your form. Add buttons to your form.
Step 6: Name your button controls	Rename your buttons to something more meaningful.
Step 7: Add dialog components to your form	Add an <code>OpenFileDialog</code> component and a <code>ColorDialog</code> component to your form.
Step 8: Write code for the show a picture button event handler	Write code by using the IntelliSense tool.
Step 9: Review, comment, and test your code	Review and test your code. Add comments as needed.
Step 10: Write code for additional buttons and a check box	Write code to make other buttons and a check box work using IntelliSense.
Step 11: Run your app and try other features	Run your app and set the background color. Try other features, such as changing colors, fonts, and borders.

There are also great, free video learning resources available to you. To learn more about programming in C#, see [C# fundamentals: Development for absolute beginners](#). To learn more about programming in Visual Basic, see [Visual Basic fundamentals: Development for absolute beginners](#).

Next steps

To begin the tutorial, start with [Step 1: Create a Windows Forms application project](#).

See also

- [More C# tutorials](#)
- [Visual Basic tutorials](#)
- [C++ tutorials](#)

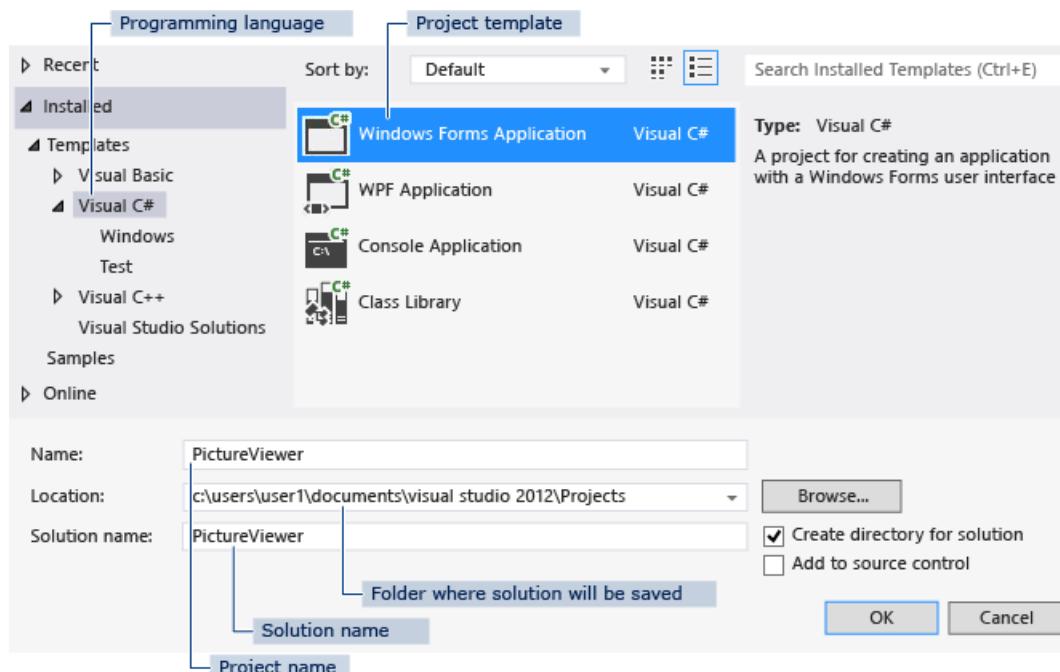
Step 1: Create a Windows Forms App project

10/16/2019 • 3 minutes to read • [Edit Online](#)

When you create a picture viewer, the first step is to create a Windows Forms App project.

Open Visual Studio 2017

1. On the menu bar, choose **File > New > Project**. The dialog box should look similar to the following screenshot.

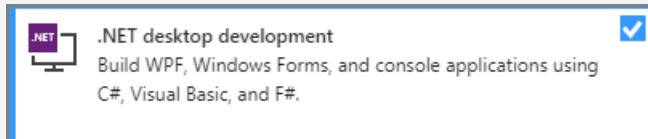


New project dialog box

2. On the left side of the **New Project** dialog box, choose either **Visual C#** or **Visual Basic**, and then choose **Windows Desktop**.
3. In the project templates list, choose **Windows Forms App (.NET Framework)**. Name the new form *PictureViewer*, and then choose the **OK** button.

NOTE

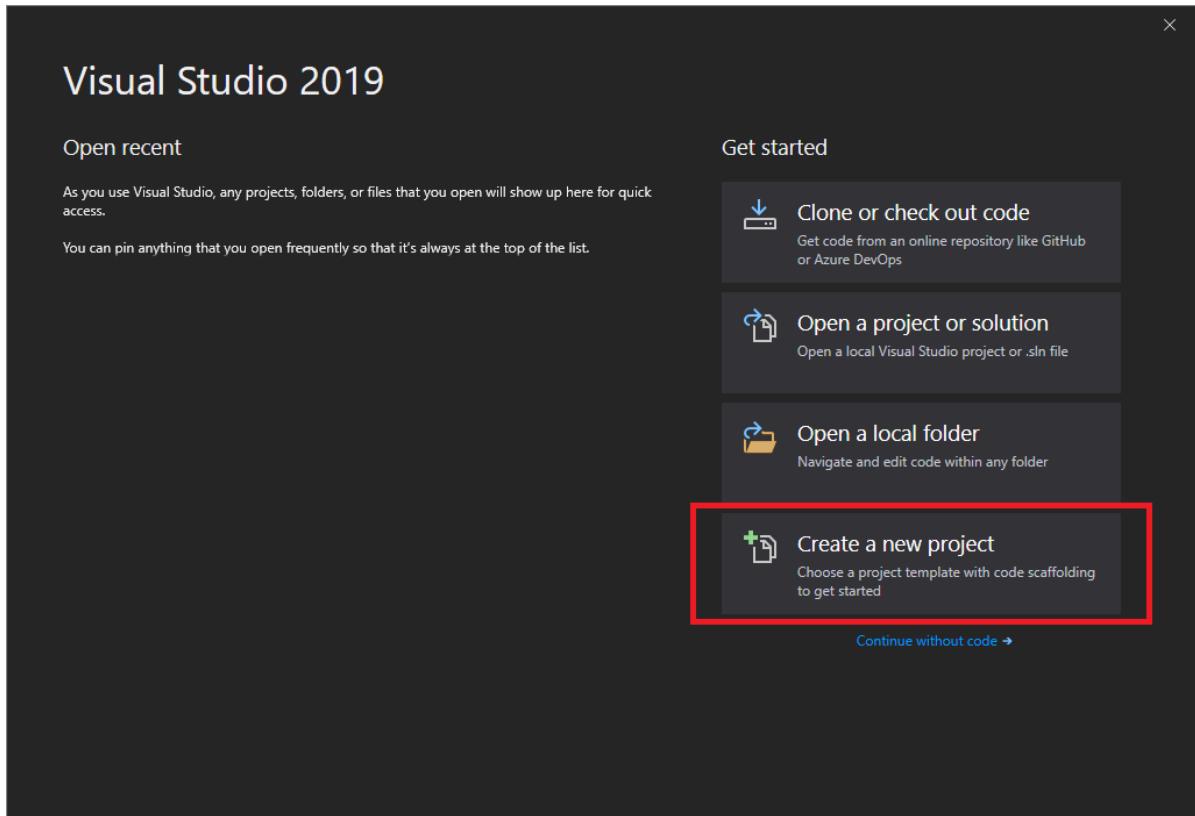
If you don't see the **Windows Forms App (.NET Framework)** template, use the Visual Studio Installer to install the **.NET desktop development** workload.



For more information, see the [Install Visual Studio](#) page.

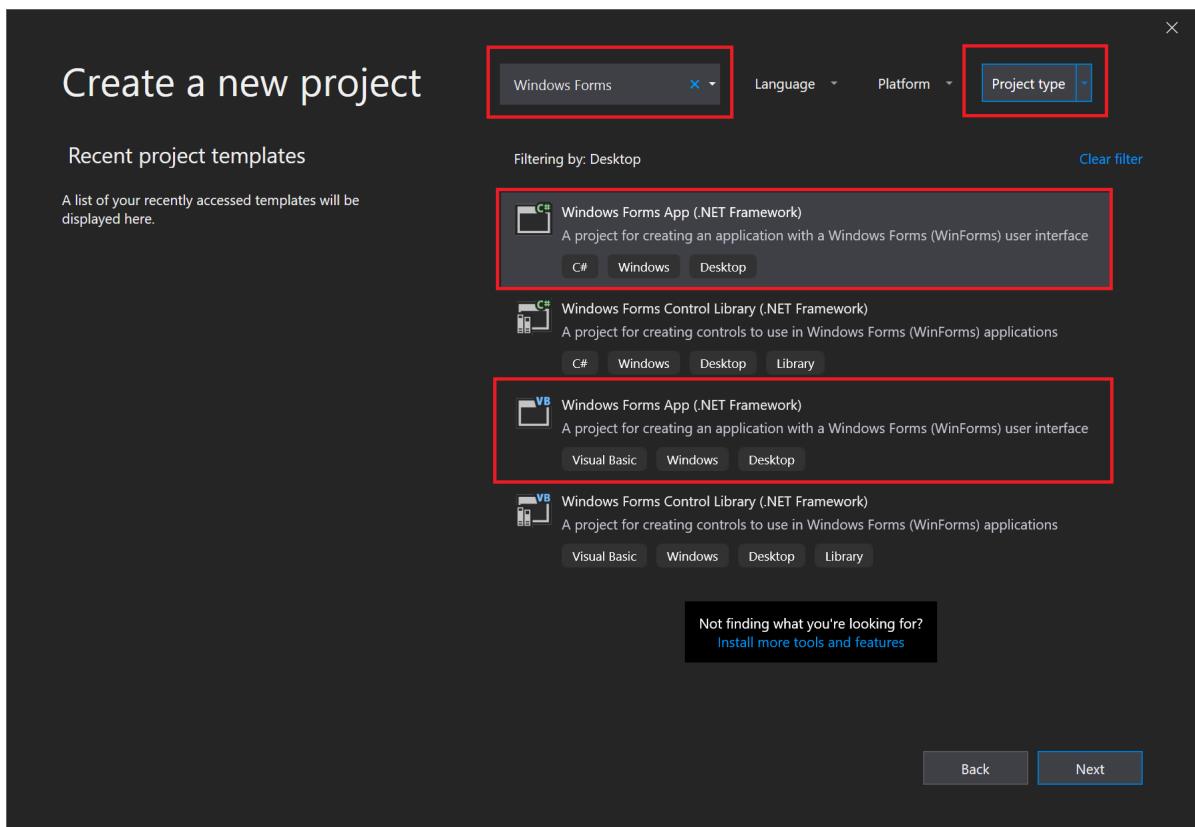
Open Visual Studio 2019

1. On the start window, choose **Create a new project**.



2. On the **Create a new project** window, enter or type *Windows Forms* in the search box. Next, choose **Desktop** from the **Project type** list.

After you apply the **Project type** filter, choose the **Windows Forms App (.NET Framework)** template for either C# or Visual Basic, and then choose **Next**.

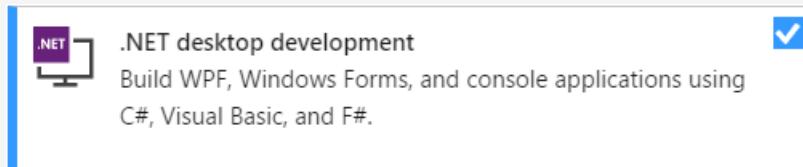


NOTE

If you don't see the **Windows Forms App (.NET Framework)** template, you can install it from the **Create a new project** window. In the **Not finding what you're looking for?** message, choose the **Install more tools and features** link.

Not finding what you're looking for?
[Install more tools and features](#)

Next, in the Visual Studio Installer, choose the Choose the **.NET desktop development** workload.



After that, choose the **Modify** button in the Visual Studio Installer. You might be prompted to save your work; if so, do so. Next, choose **Continue** to install the workload.

3. In the **Configure your new project** window, type or enter *PictureViewer* in the **Project name** box. Then, choose **Create**.

Visual Studio creates a solution for your app. A solution acts as a container for all of the projects and files needed by your app. These terms will be explained in more detail later in this tutorial.

About the Windows Forms App project

1. The development environment contains three windows: a main window, **Solution Explorer**, and the **Properties** window.

If any of these windows are missing, you can restore the default window layout. On the menu bar, choose **Window > Reset Window Layout**.

You can also display windows by using menu commands. On the menu bar, choose **View > Properties Window** or **Solution Explorer**.

If any other windows are open, close them by choosing the **Close** (x) button in their upper-right corners.

- **Main window** In this window, you'll do most of your work, such as working with forms and editing code. The window shows a form in the **Form Editor**. At the top of the window, the **Start Page** tab and the **Form1.cs [Design]** tab appear. (In Visual Basic, the tab name ends with **.vb** instead of **.cs**.)
- **Main window** In this window, you'll do most of your work, such as working with forms and editing code. The window shows a form in the **Form Editor**.
- **Solution Explorer window** In this window, you can view and navigate to all items in your solution.

If you choose a file, the contents of the **Properties** window changes. If you open a code file (which ends in **.cs** in C# and **.vb** in Visual Basic), the code file or a designer for the code file appears. A designer is a visual surface onto which you can add controls such as buttons and lists. For Visual Studio forms, the designer is called the **Windows Forms Designer**.

- **Properties window** In this window, you can change the properties of items that you choose in the other windows. For example, if you choose Form1, you can change its title by setting the **Text** property, and you can change the background color by setting the **Backcolor** property.

NOTE

The top line in **Solution Explorer** shows **Solution 'PictureViewer' (1 project)**, which means that Visual Studio created a solution for you. A solution can contain more than one project, but for now, you'll work with solutions that contain only one project.

2. On the menu bar, choose **File > Save All**.

As an alternative, choose the **Save All** button on the toolbar, which the following image shows.



Save All toolbar button

Visual Studio automatically fills in the folder name and the project name and then saves the project in your projects folder.

Next steps

- To go to the next tutorial step, see [Step 2: Run your app](#).
- To return to the overview topic, see [Tutorial 1: Create a picture viewer](#).

See also

- [Tutorial 2: Create a timed math quiz](#)
- [Tutorial 3: Create a matching game](#)

Step 2: Run your picture viewer app

10/3/2019 • 2 minutes to read • [Edit Online](#)

When you create a Windows Forms App project, you actually build a program that runs. In this tutorial, your picture viewer app doesn't do much yet—although it will. For now, it displays an empty window that shows **Form1** in the title bar.

Here's how to run your app.

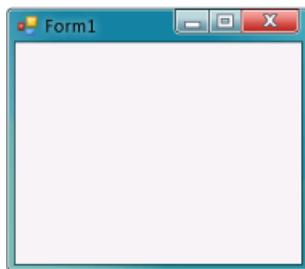
1. Choose one of the following methods:

- Choose the **F5** key.
- On the menu bar, choose **Debug > Start Debugging**.
- On the toolbar, choose the **Start Debugging** button, which appears as follows:



Start Debugging toolbar button

2. Visual Studio runs your app, and a window called **Form1** appears. The following screenshot shows the app you just built. The app is running, and you'll soon add to it.



Windows Forms App, running

3. Go back to the Visual Studio integrated development environment (IDE), and then look at the new toolbar. Additional buttons appear on the toolbar when you run an application. These buttons let you do things like stop and start your app, and help you track down any errors (bugs) it may have. For this example, we're using it to start and stop the app.



Debugging toolbar

4. Use one of the following methods to stop your app:

- On the toolbar, choose the **Stop Debugging** button.
- On the menu bar, choose **Debug > Stop Debugging**.
- Use your keyboard and press **Shift+F5**.
- Choose the **X** button in the upper corner of the **Form1** window.

NOTE

When you run your app from inside the IDE, it's called debugging because you typically do so to locate and fix bugs (errors) in the application. Although this app is small and doesn't do much yet, it's still a real program. You follow the same procedure to run and debug other programs. To learn more about debugging, see [First look at the debugger](#).

Next steps

- To go to the next tutorial step, see [Step 3: Set your form properties](#).
- To return to the previous tutorial step, see [Step 1: Create a Windows Forms App project](#).

See also

- [Tutorial 2: Create a timed math quiz](#)
- [Tutorial 3: Create a matching game](#)

Step 3: Set your form properties

9/19/2019 • 2 minutes to read • [Edit Online](#)

Next, you use the **Properties** window to change the way your form looks.

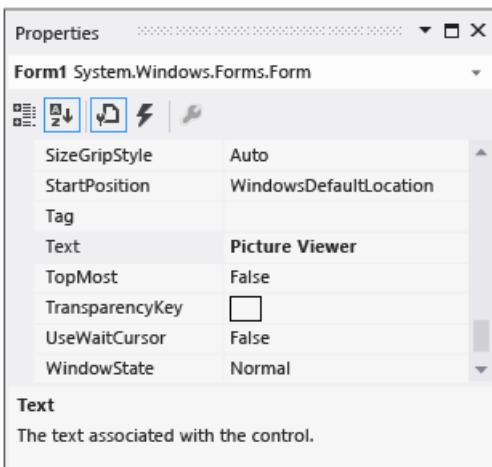
How to set your form properties

1. Be sure you're looking at **Windows Forms Designer**. In the Visual Studio integrated development environment (IDE), choose the **Form1.cs [Design]** tab (or the **Form1.vb [Design]** tab in Visual Basic).
2. Choose anywhere inside the form **Form1** to select it. Look at the **Properties** window, which should now be showing the properties for the form. Forms have various properties. For example, you can set the foreground and background color, title text that appears at the top of the form, size of the form, and other properties.

NOTE

If the **Properties** window doesn't appear, stop your app by choosing the square **Stop Debugging** button on the toolbar, or just close the window. If the app is stopped and you still don't see the **Properties** window, on the menu bar, choose **View > Properties Window**.

3. After the form is selected, find the **Text** property in the **Properties** window. Depending on how the list is sorted, you might need to scroll down. Choose **Text**, type **Picture Viewer**, and then choose **Enter**. Your form should now have the text **Picture Viewer** in its title bar, and the **Properties** window should look similar to the following screenshot.

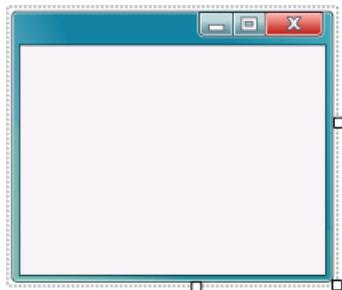


Properties window

NOTE

Properties can be ordered by a **Categorized** or **Alphabetical** view. You can switch between these two views by using the buttons on the **Properties** window. In this tutorial, it's easier to find properties through the **Alphabetical** view.

4. Go back to **Windows Forms Designer**. Choose the form's lower-right drag handle, which is the small white square in the lower-right of the form and appears as follows.



Drag handle

Drag the handle to resize the form so the form is wider and a bit taller.

5. Look at the **Properties** window, and notice that the **Size** property has changed. The **Size** property changes each time you resize the form. Try dragging the form's handle to resize it to a form size of approximately **550, 350** (no need to be exact), which should work well for this project. As an alternative, you can enter the values directly in the **Size** property and then choose the **Enter** key.
6. Run your app again. Remember, you can use any of the following methods to run your app.

- Choose the **F5** key.
- On the menu bar, choose **Debug > Start Debugging**.
- On the toolbar, choose the **Start Debugging** button, which appears as follows.



Start Debugging toolbar button

Just like before, the IDE builds and runs your app, and a window appears.

7. Before going to the next step, stop your app, because the IDE won't let you change your app while it's running. Remember, you can use any of the following methods to stop your app.

 - On the toolbar, choose the **Stop Debugging** button.
 - On the menu bar, choose **Debug > Stop Debugging**.
 - Use your keyboard and press **Shift+F5**.
 - Choose the **X** button in the upper corner of the **Picture Viewer** window.

Next steps

- To go to the next tutorial step, see [Step 4: Lay out your form with a TableLayoutPanel control](#).
- To return to the previous tutorial step, see [Step 2: Run your picture viewer app](#).

See also

- [Tutorial 2: Create a timed math quiz](#)
- [Tutorial 3: Create a matching game](#)

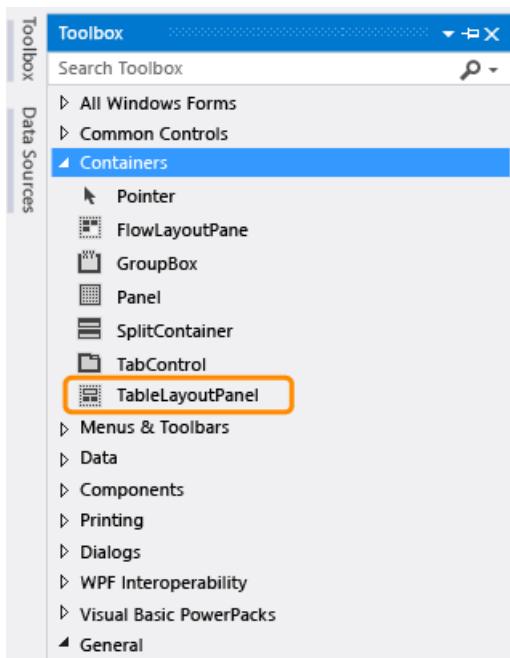
Step 4: Lay out your form with a TableLayoutPanel control

9/19/2019 • 4 minutes to read • [Edit Online](#)

In this step, you add a **TableLayoutPanel** control to your form. The TableLayoutPanel helps properly align controls in the form that you'll add later.

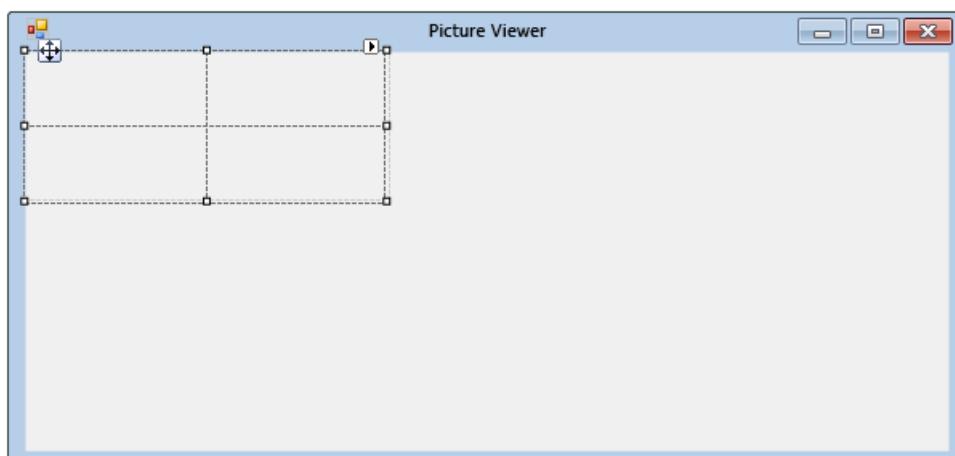
How to lay out your form with a TableLayoutPanel control

1. On the left side of the Visual Studio IDE, choose the **Toolbox** tab. (Alternatively, choose **View > Toolbox** from the menu bar, or press **Ctrl+Alt+X**.)
2. Choose the small triangle symbol next to the **Containers** group to open it, as shown in the following screenshot.



Containers group

3. You can add controls like buttons, check boxes, and labels to your form. Double-click the TableLayoutPanel control in the **Toolbox**. (Or, you can drag the control from the toolbox onto the form.) When you do this, the IDE adds a TableLayoutPanel control to your form, as shown in the following screenshot.



TableLayoutPanel control

NOTE

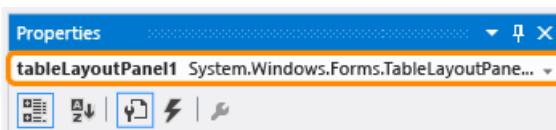
After you add your TableLayoutPanel, if a window appears inside your form with the title **TableLayoutPanel Tasks**, choose anywhere inside the form to close it. You'll learn more about this window later in the tutorial.

Notice how the **Toolbox** expands to cover your form when you choose its tab, and closes after you choose anywhere outside of it. That's the Auto Hide feature in the IDE. You can turn it on or off for any of the windows by choosing the pushpin icon in the upper-right corner of the window to toggle Auto Hide and lock it in place. The pushpin icon appears as follows.



Pushpin icon

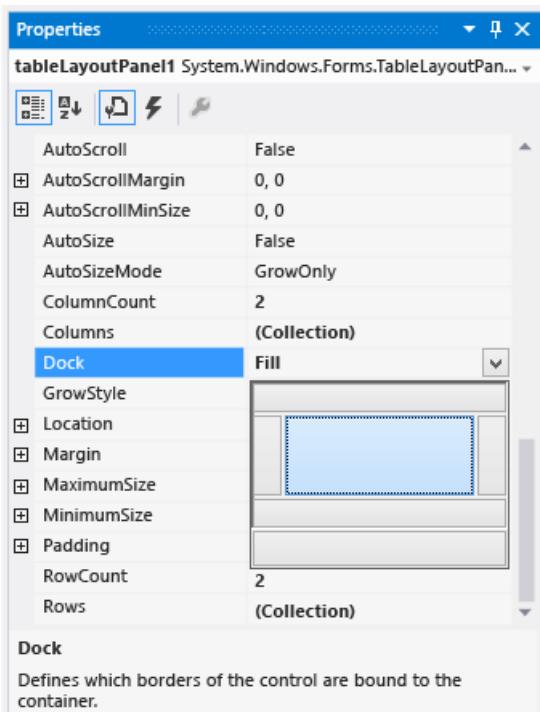
4. Be sure the TableLayoutPanel is selected by choosing it. You can verify what control is selected by looking at the drop-down list at the top of the **Properties** window, as shown in the following screenshot.



Properties window showing **TableLayoutPanel** control

5. Choose the **Alphabetical** button on the toolbar in the **Properties** window. This sorts the list of properties in the **Properties** window in alphabetical order, which makes it easier to locate properties in this tutorial.
6. The control selector is a drop-down list at the top of the **Properties** window. In this example, it shows that a control called `tableLayoutPanel1` is selected. You can select controls either by choosing an area in **Windows Forms Designer** or by choosing from the control selector.

Now that the TableLayoutPanel is selected, find the **Dock** property and choose **Dock**, which should be set to **None**. Notice that a drop-down arrow appears next to the value. Choose the arrow, and then select the **Fill** button (the large button in the middle), as shown in the following screenshot.



Properties window with **Fill** selected

Docking in Visual Studio refers to when a window is attached to another window or area in the IDE. For

example, the **Properties** window can be undocked—that is, unattached and free-floating within Visual Studio—or it can be docked against **Solution Explorer**.

7. After you set the **TableLayoutPanel Dock** property to **Fill**, notice that the panel fills the entire form. If you resize the form again, the **TableLayoutPanel** stays docked, and resizes itself to fit.

NOTE

A **TableLayoutPanel** works like a table in Microsoft Office Word: It has rows and columns, and an individual cell can span multiple rows and columns. Each cell can hold one control (like a button, a check box, or a label). Your **TableLayoutPanel** should have a **PictureBox** control spanning its entire top row, a **CheckBox** control in its lower-left cell, and four **Button** controls in its lower-right cell.

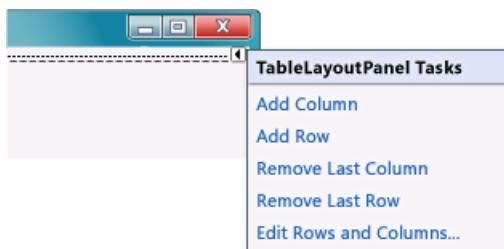
8. Currently, the **TableLayoutPanel** has two equal-size rows and two equal-size columns. Let's resize them so the top row and right column are both much bigger. In **Windows Forms Designer**, select the **TableLayoutPanel**. In the upper-right corner, there is a small black triangle button, which appears as follows.



Triangle button

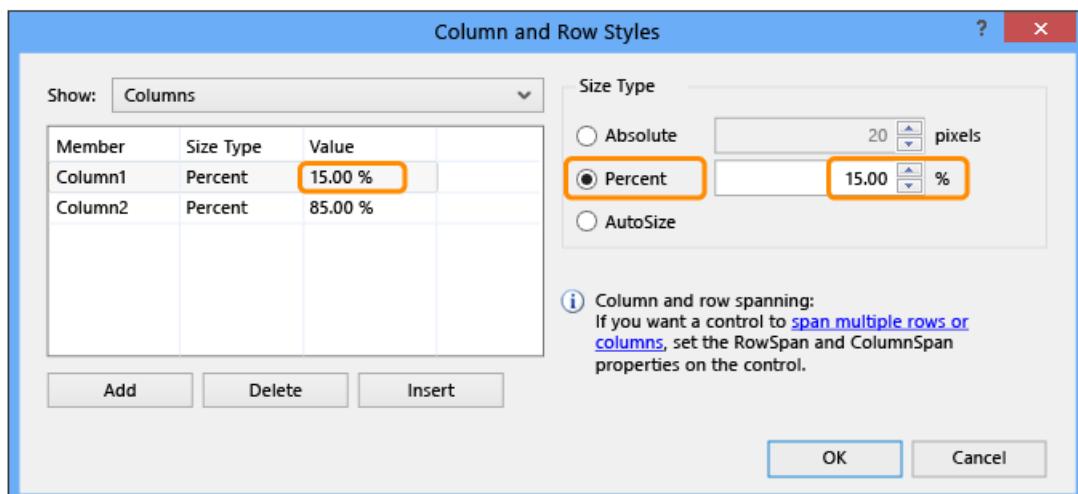
This button indicates that the control has tasks that help you set its properties automatically.

9. Choose the triangle to display the control's task list, as shown in the following screenshot.



TableLayoutPanel tasks

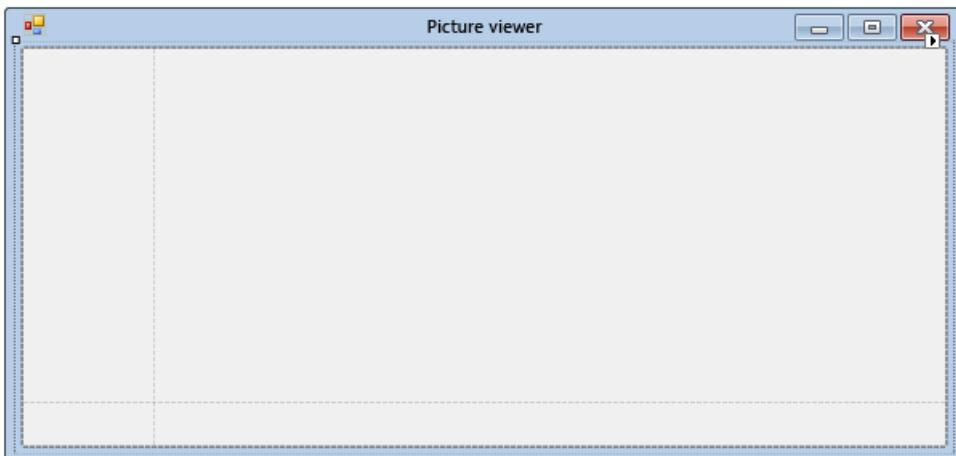
10. Choose the **Edit Rows and Columns** task to display the **Column and Row Styles** window. Choose **Column1**, and set its size to 15 percent by being sure the **Percent** button is selected and entering **15** in the **Percent** box. (That's a **NumericUpDown** control, which you'll use in a later tutorial.) Choose **Column2** and set it to 85 percent. Don't choose the **OK** button yet, because the window will close. (But if you do, you can reopen it by using the task list.)



TableLayoutPanel column and row styles

11. From the **Show** drop-down list at the top of the **Column and Row Styles** window, choose **Rows**. Set **Row1** to 90 percent and **Row2** to 10 percent.

12. Choose the **OK** button. Your TableLayoutPanel should now have a large top row, a small bottom row, a small left column, and a large right column. (You can resize the rows and columns in the TableLayoutPanel by choosing **tableLayoutPanel1** in the form and then dragging its row and column borders.)



Form 1 (Picture Viewer) with resized TableLayoutPanel

Next steps

- To go to the next tutorial step, see [Step 5: Add controls to your form](#).
- To return to the previous tutorial step, see [Step 3: Set your form properties](#).

See also

- [Tutorial 2: Create a timed math quiz](#)
- [Tutorial 3: Create a matching game](#)

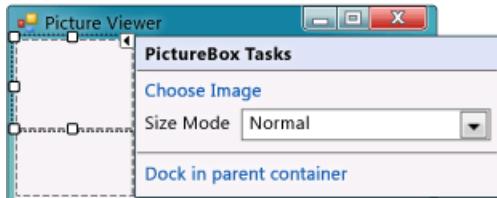
Step 5: Add controls to your form

9/19/2019 • 5 minutes to read • [Edit Online](#)

In this step, you add controls, such as a **PictureBox** control and a **CheckBox** control, to your form. You then add **Button** controls to your form.

How to add controls to your form

1. Choose the **Toolbox** tab on the left side of the Visual Studio IDE (or press **Ctrl+Alt+X**), and then expand the **Common Controls** group. This shows the most common controls that you see on forms.
2. Double-click the **PictureBox** item to add a PictureBox control to your form. Because the TableLayoutPanel is docked to fill your form, the IDE adds the PictureBox control to the first empty cell (the upper left corner).
3. Choose the new **PictureBox** control to select it, and then choose the black triangle on the new PictureBox control to display its task list, as shown in the following screenshot.



PictureBox tasks

NOTE

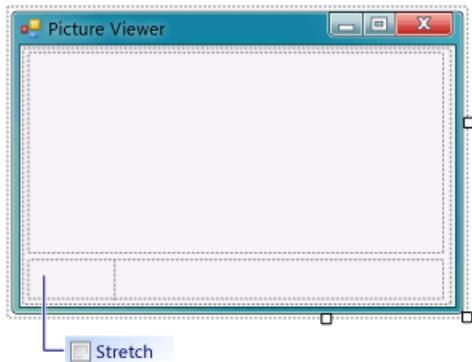
If you accidentally add the wrong type of control to your TableLayoutPanel, you can delete it. Right-click the control, and then choose **Delete** on its context menu. You can also remove controls from the form by using the menu bar. On the menu bar, choose **Edit > Undo**, or **Edit > Delete**.

4. In the **PictureBox Tasks** menu from the **PictureBox** control, choose the **Dock in parent container** link. This automatically sets the PictureBox **Dock** property to **Fill**. To see this, choose the **PictureBox** control to select it, go to the **Properties** window, and be sure that the **Dock** property is set to **Fill**.
5. Make the PictureBox span both columns by changing its **ColumnSpan** property. In the **PictureBox**, choose the **PictureBox** control and set its **ColumnSpan** property to **2**. Also, when the PictureBox is empty, you want to show an empty frame. Set its **BorderStyle** property to **Fixed3D**.

NOTE

If you don't see a **ColumnSpan** property for your PictureBox, then it's likely that the PictureBox was added to the form rather than the TableLayoutPanel. To fix this, choose the **PictureBox**, delete it, choose the **TableLayoutPanel**, and then add a new PictureBox.

6. Choose the **TableLayoutPanel** on the form and then add a CheckBox control to the form. Double-click the **CheckBox** item in the **Toolbox** to add a new CheckBox control to the next free cell in your table. Because a PictureBox takes up the first two cells in the TableLayoutPanel, the CheckBox control is added to the lower-left cell. Choose the **Text** property and type in the word **Stretch**, as shown in the following image.



TextBox control with **Stretch** property

7. Choose the **TableLayoutPanel** on the form, and then go to the **Containers** group in the **Toolbox** (where you got your TableLayoutPanel control) and double-click the **FlowLayoutPanel** item to add a new control to the last cell (bottom right). Then, dock the FlowLayoutPanel in the TableLayoutPanel. You can do so either by choosing **Dock in parent container** on the FlowLayoutPanel's black triangle task list, or by setting the FlowLayoutPanel's **Dock** property to **Fill**.

NOTE

A **FlowLayoutPanel** is a container that arranges other controls in a row, one after another. When you resize a FlowLayoutPanel, it lays out all of its controls in a single row, if it has room to do so. Otherwise, it arranges them in lines, one on top of the other.

Here, you'll use a FlowLayoutPanel to hold four buttons. If the buttons arrange one on top another when you add them, make sure that you select the FlowLayoutPanel before you add the buttons.

(Typically, each cell contains only one control. In this example, the lower-right cell of the TableLayoutPanel contains four button controls. Why? Because the FlowLayoutPanel is a container control, which is a control in a cell that holds other controls.)

To add buttons

1. Choose the new FlowLayoutPanel that you added. Go to **Common Controls** in the **Toolbox** and double-click the **Button** item to add a button control called **button1** to your FlowLayoutPanel. Repeat to add another button. The IDE determines that there's already a button called **button1** and calls the next one **button2**.
2. Typically, you add the other buttons by using the **Toolbox**. This time, choose **button2**, and then from the menu bar, choose **Edit > Copy** (or press **Ctrl+C**). Next, choose **Edit > Paste** from the menu bar (or press **Ctrl+V**) to paste a copy of your button. Now paste it again. Notice that the IDE adds **button3** and **button4** to the FlowLayoutPanel.

NOTE

You can copy and paste any control. The IDE names and places the new controls in a logical manner. If you paste a control into a container, the IDE chooses the next logical space for placement.

3. Choose the first button and set its **Text** property to **Show a picture**. Then set the **Text** properties of the next three buttons to **Clear the picture**, **Set the background color**, and **Close**.
4. Let's size the buttons and arrange them so they align to the right side of the panel. Choose the **FlowLayoutPanel** and look at its **FlowDirection** property. Change it so it's set to **RightToLeft**.

The buttons should align themselves to the right side of the cell, and reverse their order so that the **Show a**

picture button is on the right.

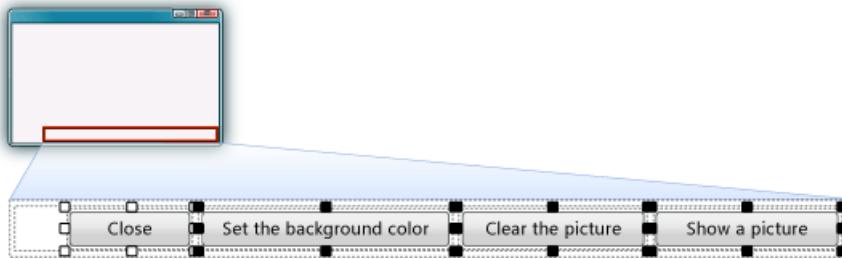
NOTE

If the buttons are still in the wrong order, you can drag the buttons around the FlowLayoutPanel to rearrange them in any order. You can choose a button and drag it left or right.

5. Choose the **Close** button to select it. Then, to choose the rest of the buttons at the same time, press and hold the **Ctrl** key and choose them, too.

After you've selected all the buttons, go to the **Properties** window and scroll up to the **AutoSize** property. This property tells the button to automatically resize itself to fit all of its text. Set it to **True**.

Your buttons should now be sized properly and be in the right order. (As long as all four buttons are selected, you can change all four **AutoSize** properties at the same time.) The following image shows the four buttons.



Picture Viewer with four buttons

6. Now run your program again to see your changes.

Notice that the buttons and the check box don't do anything yet—but they will, soon.

To continue or review

- To go to the next tutorial step, see [Step 6: Name your button controls](#).
- To return to the previous tutorial step, see [Step 4: Lay out your form with a TableLayoutPanel control](#).

See also

- [Tutorial 2: Create a timed math quiz](#)
- [Tutorial 3: Create a matching game](#)

Step 6: Name your button controls

10/15/2019 • 5 minutes to read • [Edit Online](#)

There's only one **PictureBox** on your form. When you added it, the IDE automatically named it **pictureBox1**.

There's only one **CheckBox**, which is named **checkBox1**. Soon, you'll write some code, and that code will refer to the CheckBox and PictureBox. Because there's only one of each of these controls, you'll know what it means when you see **pictureBox1** or **checkBox1** in your code.

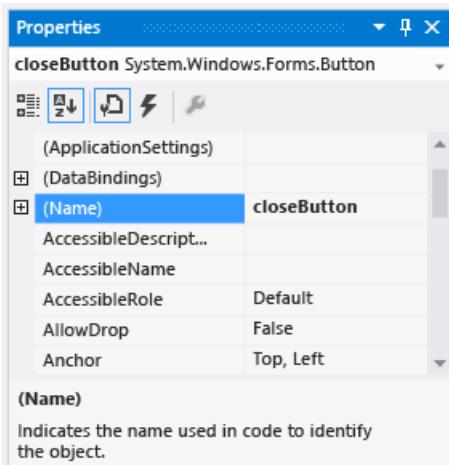
TIP

In Visual Basic, the default first letter of any control name is initial cap, so the names are **PictureBox1**, **CheckBox1**, and so on.

There are four buttons on your form, and the IDE named them **button1**, **button2**, **button3**, and **button4**. By just looking at their current names, you don't know which button is the **Close** button and which one is the **Show a picture** button. That's why giving your button controls more informative names is helpful.

To name your button controls

1. On the form, choose the **Close** button. (If you still have all the buttons selected, choose the **Esc** key to cancel the selection.) Scroll in the **Properties** window until you see the **(Name)** property. (The **(Name)** property is near the top when the properties are alphabetical.) Change the name to **closeButton**, as shown in the following screenshot.



Properties window with **closeButton** name

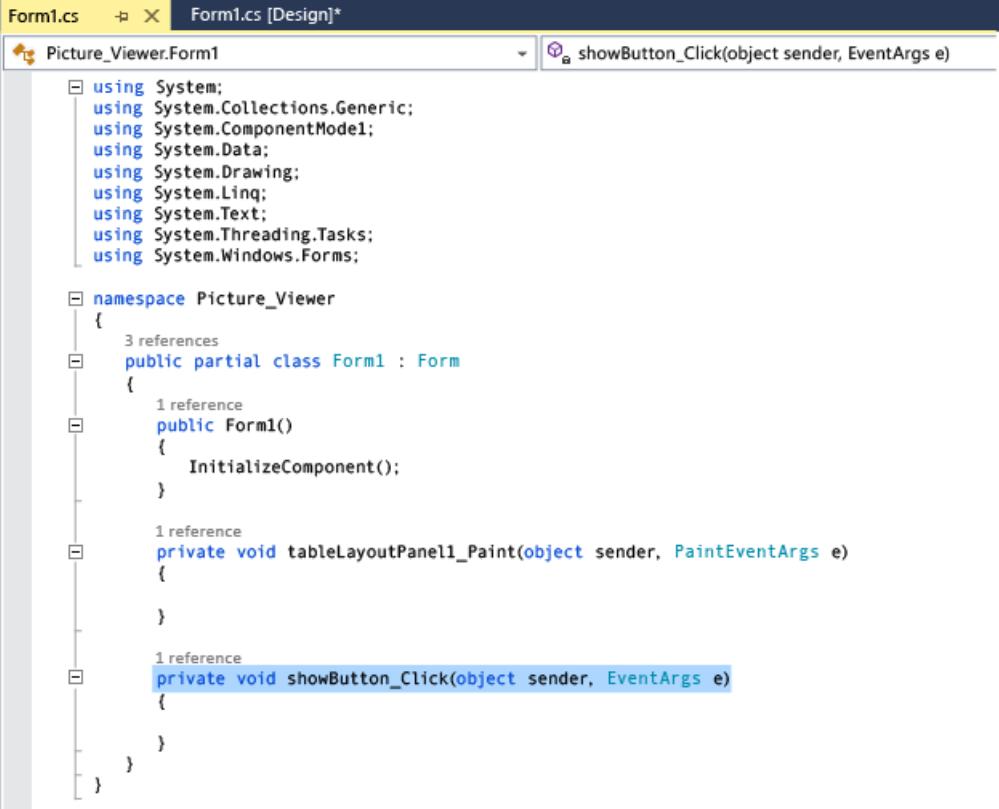
NOTE

Try changing the name of your button to **close Button**, with a space between the words "close" and "Button". When you do so, the IDE displays an error message: "Property value is not valid." Spaces (and a few other characters) are not allowed in control names.

2. Rename the other three buttons to **backgroundButton**, **clearButton**, and **showButton**. You can verify the names by choosing the control selector drop-down list in the **Properties** window. The new button names appear.
3. Double-click the **Show a picture** button on the form. As an alternative, choose the **Show a picture** button

on the form, and then press the **Enter** key. When you do, the IDE opens an additional tab in the main window named **Form1.cs**. (If you're using Visual Basic, the tab is named **Form1.vb**).

This tab displays the code file behind the form, as shown in the following screenshot.



The screenshot shows the Visual Studio IDE with the 'Form1.cs' tab selected. The code editor displays the following C# code:

```
Form1.cs  Form1.cs [Design]*

Picture_Viewer.Form1
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace Picture_Viewer
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        private void tableLayoutPanel1_Paint(object sender, PaintEventArgs e)
        {

        }

        private void showButton_Click(object sender, EventArgs e)
        {
        }
    }
}
```

Form1.cs tab with C# code

NOTE

Your Form1.cs or Form1.vb tab might display **showButton** as **ShowButton** instead.

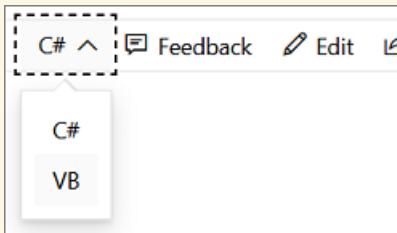
4. Focus on this part of the code.

```
private void ShowButton_Click(object sender, EventArgs e)
{}
```

```
Private Sub showButton_Click() Handles showButton.Click
End Sub
```

IMPORTANT

Use the programming language control at the top right of this page to view either the C# code snippet or the Visual Basic code snippet.



You're looking at code called `showButton_Click()` (alternatively, `ShowButton_Click()`). The IDE added this to the form's code when you opened the code file for the **showButton** button. At design-time, when you open the code file for a control in a form, code is generated for the control if it doesn't already exist. This code, known as a *method*, runs when you run your app and choose the control - in this case, the **Show a picture** button.

5. Choose the **Windows Forms Designer** tab again (**Form1.cs [Design]**), and then open the code file for the **Clear the picture** button to create a method for it in the form's code. Repeat this for the remaining two buttons. Each time, the IDE adds a new method to the form's code file.
6. To add one more method, open the code file for the **CheckBox** control in **Windows Forms Designer** to make the IDE add a `checkBox1_CheckedChanged()` method. That method is called whenever the user selects or clears the check box.

TIP

When working on an app, you often move between the code editor and **Windows Forms Designer**. The IDE makes it easy to navigate in your project. Use **Solution Explorer** to open **Windows Forms Designer** by double-clicking *Form1.cs* in C# or *Form1.vb* in Visual Basic, or on the menu bar, choose **View > Designer**.

The following shows the new code that you see in the code editor.

```
private void clearButton_Click(object sender, EventArgs e)
{
}

private void backgroundButton_Click(object sender, EventArgs e)
{
}

private void closeButton_Click(object sender, EventArgs e)
{
}

private void checkBox1_CheckedChanged(object sender, EventArgs e)
{}
```

```
Private Sub clearButton_Click() Handles clearButton.Click
    End Sub

    Private Sub backgroundButton_Click() Handles backgroundButton.Click
    End Sub

    Private Sub closeButton_Click() Handles closeButton.Click
    End Sub

    Private Sub CheckBox1_CheckedChanged() Handles CheckBox1.CheckedChanged
    End Sub
```

NOTE

Your code might not display event handlers in "camelCase" letters.

The five methods that you added are called *event handlers*, because your application calls them whenever an event (like a user choosing a button or selecting a box) happens.

When you view the code for a control in the IDE at design time, Visual Studio adds an event handler method for the control if one isn't there. For example, when you double-click a button, the IDE adds an event handler for its [Click](#) event (which is called whenever the user chooses the button). When you double-click a check box, the IDE adds an event handler for its [CheckedChanged](#) event (which is called whenever the user selects or clears the box).

After you add an event handler for a control, you can return to it at any time from **Windows Forms Designer** by double-clicking the control, or on the menu bar, choosing **View > Code**.

Names are important when you build programs, and methods (including event handlers) can have any name that you want. When you add an event handler with the IDE, it creates a name based on the control's name and the event being handled.

For example, the Click event for a button named **showButton** is called the `showButton_Click()` (alternatively, `ShowButton_Click()`) event handler method. Also, opening and closing parentheses `()` are usually added after the method name to indicate that methods are being discussed.

If you decide you want to change a code variable name, right-click the variable in the code and then choose **Refactor > Rename**. All instances of that variable in the code are renamed. For more information, see [Rename refactoring](#).

Next steps

- To go to the next tutorial step, see [Step 7: Add dialog components to your form](#).
- To return to the previous tutorial step, see [Step 5: Add controls to your form](#).

See also

- [Tutorial 2: Create a timed math quiz](#)
- [Tutorial 3: Create a matching game](#)

Step 7: Add dialog components to your form

9/19/2019 • 2 minutes to read • [Edit Online](#)

To enable your app to open picture files and choose a background color, in this step, you add an [OpenFileDialog](#) component and a [ColorDialog](#) component to your form.

A component is like a control in some ways. You use the **Toolbox** to add a component to your form, and you set its properties using the **Properties** window. But unlike a control, adding a component to your form doesn't add a visible item that the user can see on the form. Instead, it provides certain behaviors that you can trigger with code. It's a component that opens an **Open File** dialog box.

To add dialog components to your form

1. Choose the **Windows Forms Designer (Form1.cs [Design])**, and then open the **Dialogs** group in the **Toolbox**.

NOTE

The **Dialogs** group in the **Toolbox** has components that open many useful dialog boxes for you, which can be used for opening and saving files, browsing folders, and choosing fonts and colors. You use two dialog components in this project: OpenFileDialog and ColorDialog.

2. To add a component called **openFileDialog1** to your form, double-click **OpenFileDialog**. To add a component called **colorDialog1** to your form, double-click **ColorDialog** in the **Toolbox**. (You use that one in the next tutorial step.) You should see an area at the bottom of **Windows Forms Designer** (beneath the **Picture Viewer** form) that has an icon for each of the two dialog components that you added, as shown in the following image.



Dialog components

3. Choose the **openFileDialog1** icon in the area at the bottom of the **Windows Forms Designer**. Set two properties:

- Set the **Filter** property to the following (you can copy and paste it):

```
JPEG Files (*.jpg)|*.jpg|PNG Files (*.png)|*.png|BMP Files (*.bmp)|*.bmp>All files (*.*)|*.*
```

- Set the **Title** property to the following: **Select a picture file**

The **Filter** property settings specify the kinds of file types that will display in the **Select a picture** file dialog box.

TIP

To see an example of the **Open File** dialog box in a different application, open **Notepad** or **Paint**, and on the menu bar, choose **File > Open**. Notice how there's a drop-down list next to the file name that lets you choose the file type.

You just used the **Filter** property in the **OpenFileDialog** component to set that up in your app. Also, notice how the **Title** and **Filter** properties are bold in the **Properties** window. The IDE does that to show you any properties that have been changed from their default values.

Next steps

- To go to the next tutorial step, see [Step 8: Write code for the show a picture button event handler](#).
- To return to the previous tutorial step, see [Step 6: Name your button controls](#).

See also

- [Tutorial 2: Create a timed math quiz](#)
- [Tutorial 3: Create a matching game](#)

Step 8: Write code for the show a picture button event handler

10/15/2019 • 5 minutes to read • [Edit Online](#)

In this step, you make the **Show a picture** button work as follows:

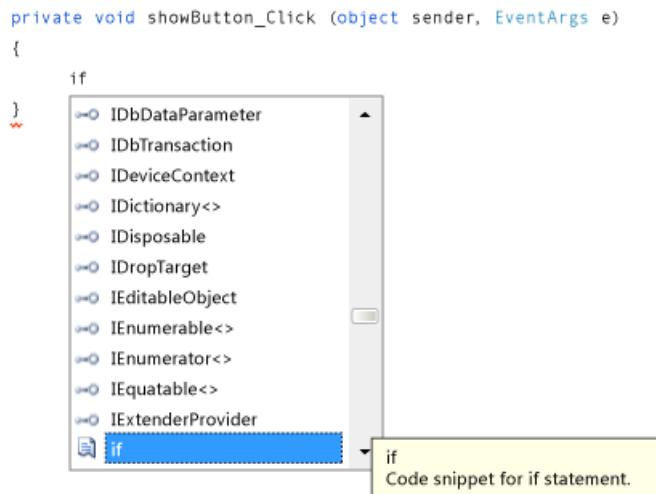
- When a user chooses that button, the app opens an [OpenFileDialog](#) box.
- If a user opens a picture file, the app shows that picture in the [PictureBox](#).

The IDE has a powerful tool called IntelliSense that helps you write code. As you type code, the IDE opens a box with suggested completions for partial words that you enter.

IntelliSense tries to determine what you want to do next, and automatically jumps to the last item you choose from the list. You can use the up or down arrows to move in the list, or you can keep typing letters to narrow the choices. When you see the choice you want, choose the **Tab** key to select it. Or, you can ignore the suggestions, if not needed.

To write code for the show a picture button event handler

1. Go to **Windows Forms Designer** and double-click the **Show a picture** button. The IDE immediately goes to the code designer and moves your cursor so it's inside the `showButton_Click()` (alternatively, `ShowButton_Click()`) method that you added previously.
2. Type an `i` on the empty line between the two braces `{ }`. (In Visual Basic, type on the empty line between `Private Sub...` and `End Sub`.) An **IntelliSense** window opens, as shown in the following image.



NOTE

Your code might not display event handlers in "camelCase" letters.

3. The **IntelliSense** window should highlight the word `if`. (If not, enter a lowercase `f`, and it will.) Notice how a *tooltip* box next to the **IntelliSense** window appears with the description, **Code snippet for if statement**. (In Visual Basic, the tooltip also states that this is a snippet, but with slightly different wording.) You want to use that snippet, so choose the **Tab** key to insert `if` into your code. Then choose the **Tab** key again to use the `if` snippet. (If you chose somewhere else and your **IntelliSense** window disappeared,

backspace over the `i` and retype it, and the **IntelliSense** window opens again.)

```
private void showButton_Click(object sender, EventArgs e)
{
    if (true)
    {
    }
}
```

Use IntelliSense to enter more code

Next, you use IntelliSense to enter more code to open an **Open File** dialog box. If the user chose the **OK** button, the **PictureBox** loads the file that the user selected. The following steps show how to enter the code, and although there are many steps, it's just a few keystrokes:

1. Start with the selected text **true** in the snippet. Type `op` to overwrite it. (In Visual Basic, you start with an initial cap, so type `Op`.)
2. The **IntelliSense** window opens and displays `openFileDialog1`. Choose the **Tab** key to select it. (In Visual Basic, it starts with an initial cap, so you see `OpenFileDialog1`. Ensure that `OpenFileDialog1` is selected.)
To learn more about `openFileDialog`, see [OpenFileDialog](#).
3. Type a period (`.`) (Many programmers call this a dot.) Because you typed a dot right after `openFileDialog1`, an **IntelliSense** window opens, filled in with all of the `OpenFileDialog` component's properties and methods. These are the same properties that appear in the **Properties** window when you choose it in **Windows Forms Designer**. You can also choose methods that tell the component to do things (like open a dialog box).

NOTE

The **IntelliSense** window can show you both properties and methods. To determine what is being shown, look at the icon on the left side of each item in the **IntelliSense** window. You see an image of a block next to each method, and an image of a wrench (or spanner) next to each property. There's also a lightning bolt icon next to each event.

Here are the icons that appear:



4. Start to type `ShowDialog` (capitalization is unimportant to IntelliSense). The `ShowDialog()` method will show the **Open File** dialog box. After the window has highlighted `ShowDialog`, choose the **Tab** key. You can also highlight "ShowDialog" and choose the **F1** key to get help for it.

To learn more about the `ShowDialog()` method, see [ShowDialog Method](#).

5. When you use a method on a control or a component (referred to as *calling a method*), you need to add parentheses. So enter opening and closing parentheses immediately after the "g" in `ShowDialog : ()`. It should now look like "openFileDialog1.ShowDialog()".

NOTE

Methods are an important part of any app, and this tutorial has shown several ways to use methods. You can call a component's method to tell it to do something, like how you called the `OpenFileDialog` component's `ShowDialog()` method. You can create your own methods to make your app do things, like the one you're building now, called the `showButton_Click()` method, which opens a dialog box and a picture when a user chooses a button.

- For C#, add a space, and then add two equal signs (==). For Visual Basic, add a space, and then use a single equal sign (=). (C# and Visual Basic use different equality operators.)
- Add another space. As soon as you do, another **IntelliSense** window opens. Start to type `DialogResult` and choose the **Tab** key to add it.

NOTE

When you write code to call a method, sometimes it returns a value. In this case, the **OpenFileDialog** component's `ShowDialog()` method returns a `DialogResult` value. `DialogResult` is a special value that tells you what happened in a dialog box. An **OpenFileDialog** component can result in the user choosing **OK** or **Cancel**, so its `ShowDialog()` method returns either `DialogResult.OK` or `DialogResult.Cancel`.

- Type a dot to open the `DialogResult` value **IntelliSense** window. Enter the letter `o` and choose the **Tab** key to insert **OK**.

To learn more about `DialogResult`, see [DialogResult](#).

NOTE

The first line of code should be complete. For C#, it should be similar to the following.

```
if (openFileDialog1.ShowDialog() == DialogResult.OK)
```

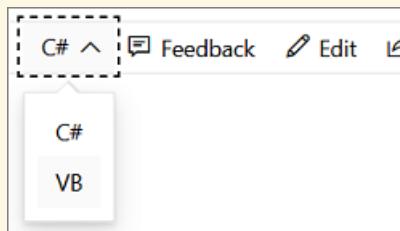
For Visual Basic, it should be the following.

```
If OpenFileDialog1.ShowDialog() = DialogResult.OK Then
```

- Now add one more line of code. You can type it (or copy and paste it), but consider using IntelliSense to add it. The more familiar you are with IntelliSense, the more quickly you can write your own code. Your final `showButton_Click()` method should look similar to the following code.

IMPORTANT

Use the programming language control at the top right of this page to view either the C# code snippet or the Visual Basic code snippet.



```
private void showButton_Click(object sender, EventArgs e)
{
    if (openFileDialog1.ShowDialog() == DialogResult.OK)
    {
        pictureBox1.Load(openFileDialog1.FileName);
    }
}
```

```
Private Sub showButton_Click() Handles showButton.Click
    If OpenFileDialog1.ShowDialog() = DialogResult.OK Then
        PictureBox1.Load(OpenFileDialog1.FileName)
    End If
End Sub
```

Next steps

- To go to the next tutorial step, see [Step 9: Review, comment, and test your code](#).
- To return to the previous tutorial step, see [Step 7: Add dialog components to your form](#).

See also

- [Tutorial 2: Create a timed math quiz](#)
- [Tutorial 3: Create a matching game](#)

Step 9: Review, comment, and test your code

10/17/2019 • 5 minutes to read • [Edit Online](#)

You next add a comment to your code. A comment is a note that doesn't change the way the app behaves. It makes it easier for someone who is reading your code to understand what it does. Adding comments to your code is a good habit to get into.

In C#, two forward slashes (//) mark a line as a comment. In Visual Basic, a single quotation mark ('') is used to mark a line as a comment. After you add a comment, you test your application. It's good practice to run and test your code frequently while you're working on your projects, so you can catch and fix any problems early, before the code gets more complicated. This is called *iterative testing*.

You just built something that works, and although it's not done yet, it can already load a picture. Before you add a comment to your code and test it, take time to review the code concepts, because you will use these concepts frequently:

- When you double-clicked the **Show a picture** button in **Windows Forms Designer**, the IDE automatically added a *method* to your program's code.
- Methods are how you organize your code: It's how your code is grouped together.
- Most of the time, a method does a small number of things in a specific order, like how your `showButton_Click()` (or `ShowButton_Click()`) method shows a dialog box and then loads a picture.
- A method is made up of code *statements*, or lines of code. Think of a method as a way to bundle code statements together.
- When a method is executed, or *called*, the statements in the method are executed in order, one after another, starting with the first one.

The following is an example of a statement.

```
PictureBox1.Load(openFileDialog1.FileName);
```

```
pictureBox1.Load(openFileDialog1.FileName)
```

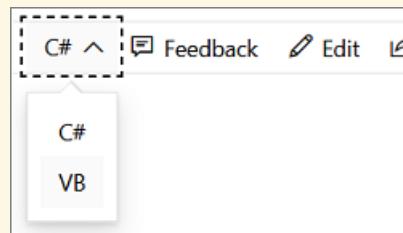
Statements are what make your programs do things. In C#, a statement always ends in a semicolon. In Visual Basic, the end of a line is the end of a statement. (No semicolon is needed in Visual Basic.) The preceding statement tells your **PictureBox** control to load the file that the user selected with the **OpenFileDialog** component.

To add comments

1. Add the following comment to your code.

IMPORTANT

Use the programming language control at the top right of this page to view either the C# code snippet or the Visual Basic code snippet.



```
private void showButton_Click(object sender, EventArgs e)
{
    // Show the Open File dialog. If the user clicks OK, load the
    // picture that the user chose.
    if (openFileDialog1.ShowDialog() == DialogResult.OK)
    {
        pictureBox1.Load(openFileDialog1.FileName);
    }
}
```

```
Private Sub showButton_Click() Handles showButton.Click

    ' Show the Open File dialog. If the user clicks OK, load the
    ' picture that the user chose.
    If OpenFileDialog1.ShowDialog() = DialogResult.OK Then
        PictureBox1.Load(OpenFileDialog1.FileName)
    End If

End Sub
```

Your **showButton** button's [Click](#) event handler is now finished, and it works. You have started writing code, starting with an `if` statement. An `if` statement is how you tell your app, "Check this one thing, and if it's true, do these actions." In this case, you tell your app to open the **Open File** dialog box, and if the user selects a file and chooses the **OK** button, load that file in the **PictureBox**.

TIP

The IDE is built to make it easy for you to write code, and *code snippets* are one way it does that. A snippet is a shortcut that gets expanded into a small block of code.

You can see all of the snippets available. On the menu bar, choose **Tools > Code Snippets Manager**. For C#, the `if` snippet is in **Visual C#**. For Visual Basic, the `if` snippets are in **Code Patterns > Conditionals and Loops**. You can use this manager to browse existing snippets or add your own snippets.

To activate a snippet when typing code, type it and choose the **Tab** key. Many snippets appear in the **IntelliSense** window, which is why you choose the **Tab** key twice: first to select the snippet from the **IntelliSense** window, and then to tell the IDE to use the snippet. (IntelliSense supports the `if` snippet, but not the `elseif` snippet.)

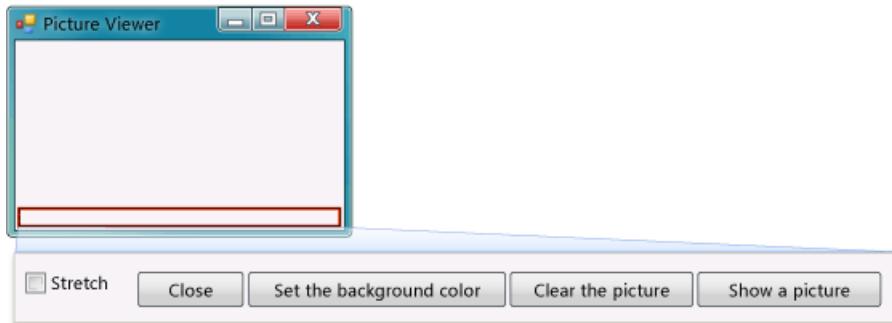
2. Before you run your application, save your app by choosing the **Save All** toolbar button, which should look similar to the following screenshot.



Save All button

Alternatively, to save your app, choose **File** > **Save All** from the menu bar (or press **Ctrl+Shift+S**). It's a best practice to save early and often.

When it's running, your program should look like the following image.



Picture Viewer

To test your app

1. Choose the **F5** key or choose the **Start Debugging** toolbar button.
2. Choose the **Show a picture** button to run the code you just wrote. First, the app opens an **Open File** dialog box. Verify that your filters appear in the **Files of type** drop-down list at the bottom of the dialog box. Then navigate to a picture and open it. You can usually find sample pictures that ship with the Windows operating system in your *My Documents* folder, inside the *My Pictures\Sample Pictures* folder.

TIP

If you don't see any images in the **Select a picture file** dialog box, be sure that the **All files (*.*)** filter is selected in the drop-down list on the lower right side of the dialog box.

3. Load a picture, and it appears in your PictureBox. Then try resizing your form by dragging its borders. Because you have your PictureBox docked inside a TableLayoutPanel, which itself is docked inside the form, your picture area will resize itself so that it's as wide as the form, and fills the top 90 percent of the form. That's why you used the **TableLayoutPanel** and **FlowLayoutPanel** containers: They keep your form sized correctly when the user resizes it.

Right now, larger pictures go beyond the borders of your picture viewer. In the next step, you'll add code to make pictures fit in the window.

To continue or review

- To go to the next tutorial step, see [Step 10: Write code for additional buttons and a check box](#).
- To return to the previous tutorial step, see [Step 8: Write code for the show a picture button event handler](#).

See also

- [Tutorial 2: Create a timed math quiz](#)
- [Tutorial 3: Create a matching game](#)

Step 10: Write code for additional buttons and a check box

10/15/2019 • 2 minutes to read • [Edit Online](#)

Now you're ready to complete the other four methods. You could copy and paste this code, but if you want to learn the most from this tutorial, type the code and use IntelliSense.

This code adds functionality to the buttons you added earlier. Without this code, the buttons don't do anything. The buttons use code in their `Click` events (and the check box uses the `CheckedChanged` event) to do different things when you activate the controls. For example, the `clearButton_Click` (or `ClearButton_Click`) event, which activates when you choose the **Clear the picture** button, erases the current image by setting its `Image` property to `null` (or, `nothing`). Each event in the code includes comments that explain what the code does.

TIP

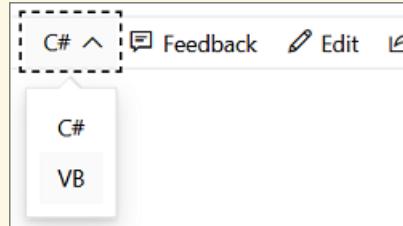
As a best practice: Always comment your code. Comments are information for a person to read, and it's worth the time to make your code understandable. Everything on a comment line is ignored by the app. In C#, you comment a line by typing two forward slashes (`//`), and in Visual Basic you comment a line by starting with a single quotation mark (`'`).

How to write code for additional buttons and a check box

Add the following code to your **Form1** code file (*Form1.cs* or *Form1.vb*).

IMPORTANT

Use the programming language control at the top right of this page to view either the C# code snippet or the Visual Basic code snippet.



```

private void clearButton_Click(object sender, EventArgs e)
{
    // Clear the picture.
    pictureBox1.Image = null;
}

private void backgroundButton_Click(object sender, EventArgs e)
{
    // Show the color dialog box. If the user clicks OK, change the
    // PictureBox control's background to the color the user chose.
    if (colorDialog1.ShowDialog() == DialogResult.OK)
        pictureBox1.BackColor = colorDialog1.Color;
}

private void closeButton_Click(object sender, EventArgs e)
{
    // Close the form.
    this.Close();
}

private void checkBox1_CheckedChanged(object sender, EventArgs e)
{
    // If the user selects the Stretch check box,
    // change the PictureBox's
    //SizeMode property to "Stretch". If the user clears
    // the check box, change it to "Normal".
    if (checkBox1.Checked)
        pictureBox1.SizeMode = PictureBoxSizeMode.StretchImage;
    else
        pictureBox1.SizeMode = PictureBoxSizeMode.Normal;
}

```

```

Private Sub clearButton_Click() Handles clearButton.Click
    ' Clear the picture.
    PictureBox1.Image = Nothing
End Sub

Private Sub backgroundButton_Click() Handles backgroundButton.Click
    ' Show the color dialog box. If the user clicks OK, change the
    ' PictureBox control's background to the color the user chose.
    If ColorDialog1.ShowDialog() = DialogResult.OK Then
        PictureBox1.BackColor = ColorDialog1.Color
    End If
End Sub

Private Sub closeButton_Click() Handles closeButton.Click
    ' Close the form.
    Close()
End Sub

Private Sub CheckBox1_CheckedChanged() Handles CheckBox1.CheckedChanged
    ' If the user selects the Stretch check box, change
    ' the PictureBox'sSizeMode property to "Stretch". If the user
    ' clears the check box, change it to "Normal".
    If CheckBox1.Checked Then
        PictureBox1.SizeMode = PictureBoxSizeMode.StretchImage
    Else
        PictureBox1.SizeMode = PictureBoxSizeMode.Normal
    End If
End Sub

```

NOTE

Your code might not display "camelCase" letters.

Next steps

- To go to the next tutorial step, see [Step 11: Run your app and try other features](#).
- To return to the previous tutorial step, see [Step 9: Review, comment, and test your code](#).

See also

- [Tutorial 2: Create a timed math quiz](#)
- [Tutorial 3: Create a matching game](#)

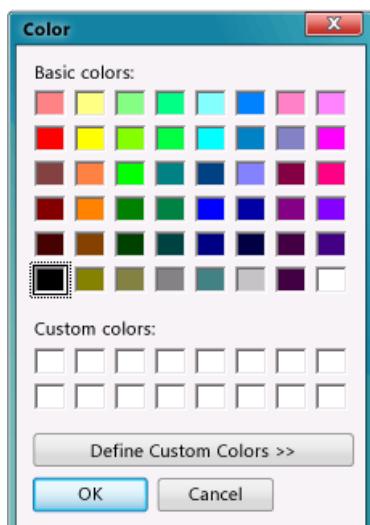
Step 11: Run your picture viewer app and try other features

9/19/2019 • 2 minutes to read • [Edit Online](#)

Your picture viewer app is finished and ready to run. You can run your app and set the background color of the **PictureBox**. To learn more, try to improve the application by changing the color of the form, customizing the buttons and check box, and changing the properties of the form.

How to run your app and set the background color

1. Choose **F5**, or on the menu bar, choose **Debug > Start Debugging**.
2. Before you open a picture, choose the **Set the background color** button. The **Color** dialog box opens.



Color dialog box

3. Choose a color to set the **PictureBox** background color. Look closely at the `backgroundButton_Click()` (or, `BackgroundButton_Click()`) method to understand how it works.

NOTE

You can load a picture from the Internet by pasting its URL into the **Open File** dialog box. Try to find an image with a transparent background, so your background color shows.

4. Choose the **Clear the picture** button to make sure it clears. Then, exit the app by choosing the **Close** button.

Try other features

- Change the color of the form and the buttons by using the **BackColor** property.
- Customize your buttons and check box using the **Font** and **ForeColor** properties.
- Change your form's **FormBorderStyle** and **ControlBox** properties.
- Use your form's **AcceptButton** and **CancelButton** properties so that buttons are automatically chosen when the user chooses the **Enter** or **Esc** key. Make the app open the **Open File** dialog box when the user

chooses **Enter** and close the box when the user chooses **Esc**.

Next steps

To learn more, continue with the following tutorial:

[Tutorial 2: Create a timed math quiz](#)

To return to the previous tutorial step, see [Step 10: Write code for additional buttons and a check box](#).

See also

- [More C# tutorials](#)
- [More Visual Basic tutorials](#)
- [C++ tutorial](#)

Tutorial 2: Create a timed math quiz

10/16/2019 • 2 minutes to read • [Edit Online](#)

In this tutorial, you build a quiz in which the quiz taker must answer four random arithmetic problems within a specified time.

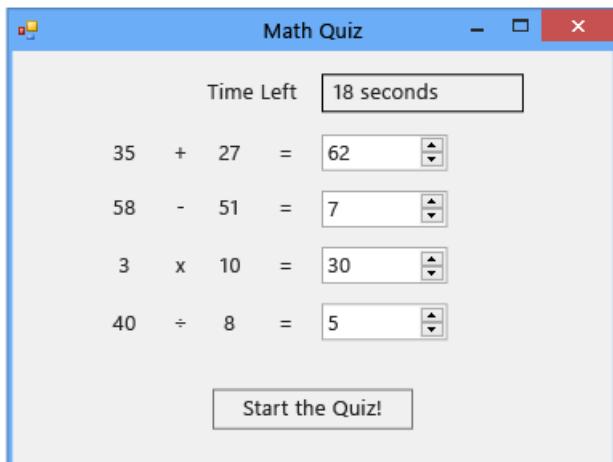
NOTE

This tutorial covers both C# and Visual Basic, so focus on the information that's specific to the programming language you're using.

This tutorial walks you through the following tasks:

- Generate random numbers by using the [Random](#) class.
- Trigger events to occur at a specific time by using a [Timer](#) control.
- Control program flow by using `if else` statements.
- Perform basic arithmetic operations in code.

When you finish, your quiz will look similar to the following screenshot, except with different numbers:



Tutorial links

TITLE	DESCRIPTION
Step 1: Create a project and add labels to your form	Start by creating the project, changing properties, and adding <code>Label</code> controls.
Step 2: Create a random addition problem	Create an addition problem, and use the <code>Random</code> class to generate random numbers.
Step 3: Add a countdown timer	Add a countdown timer so that the quiz can be timed.
Step 4: Add the CheckTheAnswer() method	Add a method to check whether the quiz taker entered a correct answer for the problem.

TITLE	DESCRIPTION
Step 5: Add Enter event handlers for the NumericUpDown controls	Add event handlers that make your quiz easier to take.
Step 6: Add a subtraction problem	Add a subtraction problem that generates random numbers, uses the timer, and checks for correct answers.
Step 7: Add multiplication and division problems	Add multiplication and division problems that generate random numbers, use the timer, and check for correct answers.
Step 8: Customize the quiz	Try other features, such as changing colors and adding a hint.

There are also great, free video learning resources available to you. To learn more about programming in C#, see [C# fundamentals: Development for absolute beginners](#). To learn more about programming in Visual Basic, see [Visual Basic fundamentals: Development for absolute beginners](#).

Next steps

To begin the tutorial, start with [Step 1: Create a project and add labels to your form](#).

See also

- [More C# tutorials](#)
- [Visual Basic tutorials](#)
- [C++ tutorials](#)

Step 1: Create a project and add labels to your form

10/16/2019 • 6 minutes to read • [Edit Online](#)

As the first steps in developing this quiz, you create the project, and you add labels, a button, and other controls to a form. You also set properties for each control that you add. The project will contain the form, the controls, and (later in the tutorial) code. The button starts the quiz, the labels show the quiz problems, and the other controls show the quiz answers and the time that remains to finish the quiz.

NOTE

This topic is part of a tutorial series about basic coding concepts.

- For an overview of the tutorial, see [Tutorial 2: Create a timed math quiz](#).
- To download a completed version of the code, see [Complete math quiz tutorial sample](#).

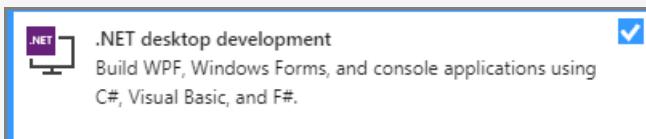
To create a project for a form

1. On the menu bar, choose **File > New > Project**.
2. Choose either **Visual C#** or **Visual Basic** on the left side of the **New Project** dialog box, and then choose **Windows Desktop**.
3. In the list of templates, choose the **Windows Forms App (.NET Framework)** template, name it *MathQuiz*, and then choose the **OK** button.

A form that's named *Form1.cs* or *Form1.vb* appears, depending on the programming language that you chose.

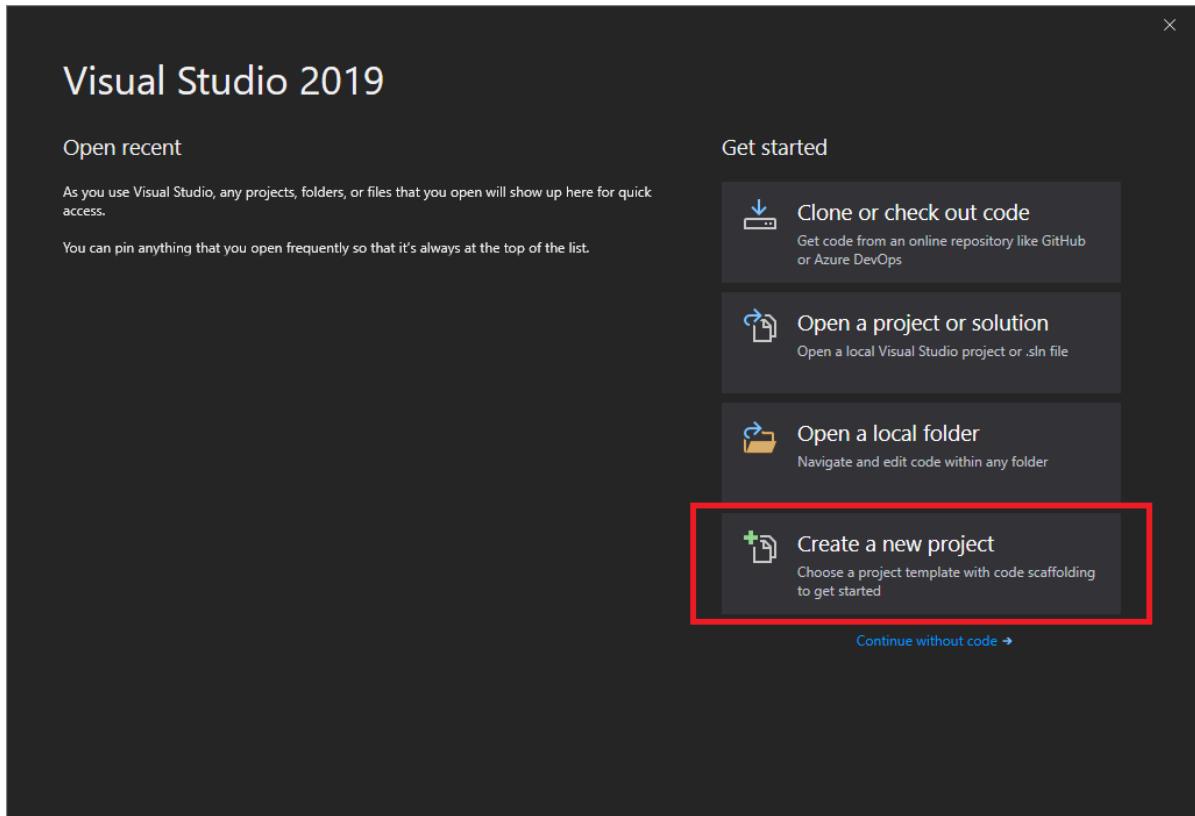
NOTE

If you don't see the **Windows Forms App (.NET Framework)** template, use the Visual Studio Installer to install the **.NET desktop development** workload.



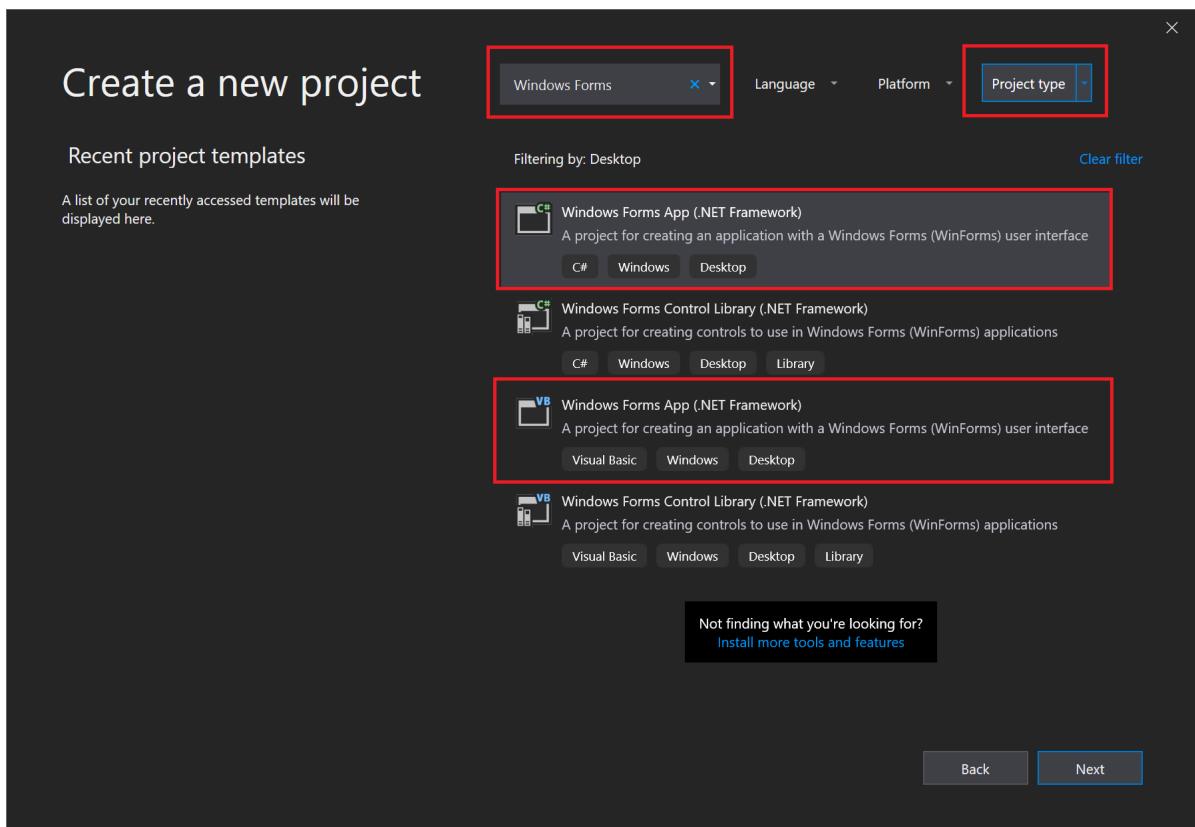
For more information, see the [Install Visual Studio](#) page.

1. On the start window, choose **Create a new project**.



2. On the **Create a new project** window, enter or type *Windows Forms* in the search box. Next, choose **Desktop** from the **Project type** list.

After you apply the **Project type** filter, choose the **Windows Forms App (.NET Framework)** template for either C# or Visual Basic, and then choose **Next**.



NOTE

If you do not see the **Windows Forms App (.NET Framework)** template, you can install it from the **Create a new project** window. In the **Not finding what you're looking for?** message, choose the **Install more tools and features** link.

Not finding what you're looking for?
[Install more tools and features](#)

Next, in the Visual Studio Installer, choose the Choose the **.NET desktop development** workload.



After that, choose the **Modify** button in the Visual Studio Installer. You might be prompted to save your work; if so, do so. Next, choose **Continue** to install the workload.

3. In the **Configure your new project** window, type or enter *MathQuiz* in the **Project name** box. Then, choose **Create**.

To set properties for a form

1. In Visual Studio, choose the form (either *Form1.cs* or *Form1.vb*, depending on the programming language), and then change its **Text** property to **Math Quiz**.

The **Properties** window contains properties for the form.

2. Change the size of the form to 500 pixels wide by 400 pixels tall.

You can resize the form by dragging its edges until the correct size appears in the lower-left corner of the integrated development environment (IDE). As an alternative, you can change the values of the **Size** property.

3. Change the value of the **FormBorderStyle** property to **Fixed3D**, and set the **MaximizeBox** property to **False**.

These values prevent quiz takers from resizing the form.

To create the time remaining box

1. Add a **Label** control from the **Toolbox**, and then set the value of its **(Name)** property to **timeLabel**.

This label will become a box in the upper-right corner that shows the number of seconds that remain in the quiz.

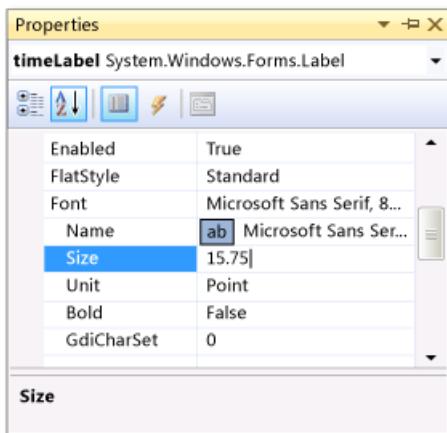
2. Change the **AutoSize** property to **False** so that you can resize the box.
3. Change the **BorderStyle** property to **FixedSingle** to draw a line around the box.
4. Set the **Size** property to **200, 30**.
5. Move the label to the upper-right corner of the form, where blue spacer lines will appear.

These lines help you align controls on the form.

6. In the **Properties** window, choose the **Text** property, and then choose the **Backspace** key to clear its value.

7. Choose the plus sign (+) next to the **Font** property, and then change the value of the **Size** property to **15.75**.

You can change several font properties, as the following screenshot shows.



8. Add another Label control from the **Toolbox**, and then set its font size to **15.75**.
9. Set the **Text** property to **Time Left**.
10. Move the label so that it lines up just to the left of the **timeLabel** label.

To add controls for the addition problems

1. Add a Label control from the **Toolbox**, and then set its **Text** property to ? (question mark).
2. Set the **AutoSize** property to **False**.
3. Set the **Size** property to **60, 50**.
4. Set the font size to **18**.
5. Set the **TextAlign** property to **MiddleCenter**.
6. Set the **Location** property to **50, 75** to position the control on the form.
7. Set the **(Name)** property to **plusLeftLabel**.
8. Choose the **plusLeftLabel** label, and then choose either the **Ctrl+C** keys or **Copy** on the **Edit** menu.
9. Paste the label three times by choosing either the **Ctrl+V** keys or **Paste** on the **Edit** menu.
10. Arrange the three new labels so that they are in a row to the right of the **plusLeftLabel** label.

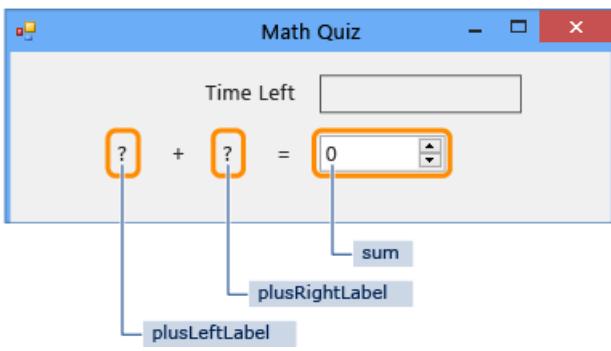
You can use the spacer lines to space them out and line them up.

11. Set the value of the second label's **Text** property to + (plus sign).
12. Set the value of the third label's **(Name)** property to **plusRightLabel**.
13. Set the value of the fourth label's **Text** property to = (equal sign).
14. Add a **NumericUpDown** control from the **Toolbox**, set its font size to **18**, and set its width to **100**.

You'll learn more about this kind of control later.

15. Line up the **NumericUpDown** control with the Label controls for the addition problem.
16. Change the value of the **(Name)** property for the **NumericUpDown** control to **sum**.

You've created the first row, as shown in the following illustration.



To add controls for the subtraction, multiplication, and division problems

1. Copy all five controls for the addition problem (the four Label controls and the NumericUpDown control), and then paste them.

The form contains five new controls, which are still selected.

2. Move all of the controls into place so that they line up below the addition controls.

You can use the spacer lines to give enough distance between the two rows.

3. Change the value of the **Text** property for the second label to - (minus sign).

4. Name the first question-mark label **minusLeftLabel**.

5. Name the second question-mark label **minusRightLabel**.

6. Name the NumericUpDown control **difference**.

7. Paste the five controls two more times.

8. For the third row, name the first label **timesLeftLabel**, change the second label's **Text** property to × (multiplication sign), name the third label **timesRightLabel**, and name the NumericUpDown control **product**.

9. For the fourth row, name the first label **dividedLeftLabel**, change the second label's **Text** property to ÷ (division sign), name the third label **dividedRightLabel**, and name the NumericUpDown control **quotient**.

NOTE

You can copy the multiplication sign × and the division sign ÷ from this tutorial and paste them onto the form.

To add a start button and set the tab-index order

1. Add a **Button** control from the **Toolbox**, and then set its **(Name)** property to **startButton**.

2. Set the **Text** property to **Start the quiz**.

3. Set the font size to **14**.

4. Set the **AutoSize** property to **True**, which causes the button to automatically resize to fit the text.

5. Center the button near the bottom of the form.

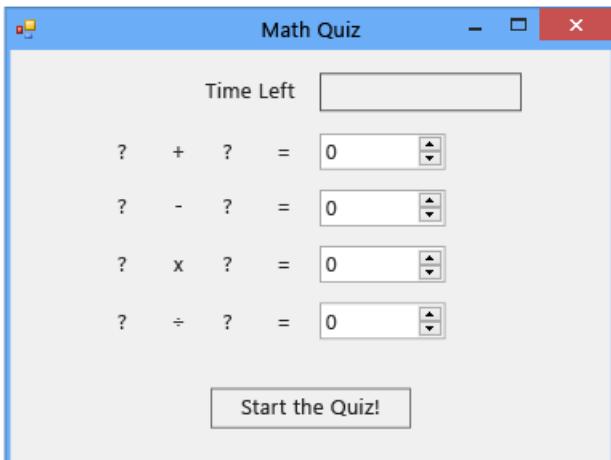
6. Set the value of the **TabIndex** property for the **startButton** control to **1**.

NOTE

The **TabIndex** property sets the order of the controls when the quiz taker chooses the **Tab** key. To see how it works, open any dialog box (for example, on the menu bar, choose **File > Open**), and then choose the **Tab** key a few times. Watch how your cursor moves from control to control each time that you choose the **Tab** key. A programmer decided the order when creating that form.

- Set the value of the **TabIndex** property for the NumericUpDown sum control to **2**, for the difference control to **3**, for the product control to **4**, and for the quotient control to **5**.

The form should look similar to the following screenshot.



- To verify whether the **TabIndex** property works as you expect, save and run your program by choosing the **F5** key, or by choosing **Debug > Start Debugging** on the menu bar, and then choose the **Tab** key a few times.

To continue or review

- To go to the next tutorial step, see [Step 2: Create a random addition problem](#).
- To return to the overview topic, see [Tutorial 2: Create a timed math quiz](#).

Step 2: Create a random addition problem

10/17/2019 • 6 minutes to read • [Edit Online](#)

In the second part of this tutorial, you make the quiz challenging by adding math problems that are based on random numbers. You also create a method that's named `StartTheQuiz()` and that fills in the problems and starts the countdown timer. Later in this tutorial, you'll add the subtraction, multiplication, and division problems.

NOTE

This topic is part of a tutorial series about basic coding concepts.

- For an overview of the tutorial, see [Tutorial 2: Create a timed math quiz](#).
- To download a completed version of the code, see [Complete math quiz tutorial sample](#).

To create a random addition problem

1. In the form designer, choose the form (**Form1**).
2. On the menu bar, choose **View > Code**.

Form1.cs or *Form1.vb* appears, depending on the programming language that you're using, so that you can view the code behind the form.

3. Create a `Random` object by adding a `new` statement near the top of the code, like the following.

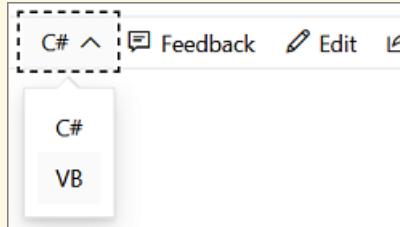
```
public partial class Form1 : Form
{
    // Create a Random object called randomizer
    // to generate random numbers.
    Random randomizer = new Random();
```

```
Public Class Form1

    ' Create a Random object called randomizer
    ' to generate random numbers.
    Private randomizer As New Random
```

IMPORTANT

Use the programming language control at the top right of this page to view either the C# code snippet or the Visual Basic code snippet.



You've added a Random object to your form and named the object **randomizer**.

`Random` is known as an object. You've probably heard that word before, and you learn more about what it means for programming in the next tutorial. For now, just remember that you can use `new` statements to create buttons, labels, panels, OpenFileDialogs, ColorDialogs, SoundPlayers, Randoms, and even forms, and those items are referred to as objects. When you run your program, the form is started, and the code behind it creates a random object and names it **randomizer**.

Soon you'll build a method to check the answers, so your quiz must use variables to store the random numbers that it generates for each problem. See [Variables](#) or [Types](#). To properly use variables, you must declare them, which means listing their names and data types.

4. Add two integer variables to the form, and name them **addend1** and **addend2**.

NOTE

An integer variable is known as an `int` in C# or an `Integer` in Visual Basic. This kind of variable stores a positive or negative number from -2147483648 through 2147483647 and can store only whole numbers, not decimals.

You use a similar syntax to add an integer variable as you did to add the random object, as the following code shows.

```
// Create a Random object called randomizer
// to generate random numbers.
Random randomizer = new Random();

// These integer variables store the numbers
// for the addition problem.
int addend1;
int addend2;
```

```
' Create a Random object called randomizer
' to generate random numbers.
Private randomizer As New Random

' These integer variables store the numbers
' for the addition problem.
Private addend1 As Integer
Private addend2 As Integer
```

5. Add a method that's named `StartTheQuiz()` and that uses the Random object's `Next()` method to show the random numbers in the labels. `StartTheQuiz()` will eventually fill in all of the problems and then start the timer, so add a comment. The function should look like the following.

```

/// <summary>
/// Start the quiz by filling in all of the problems
/// and starting the timer.
/// </summary>
public void StartTheQuiz()
{
    // Fill in the addition problem.
    // Generate two random numbers to add.
    // Store the values in the variables 'addend1' and 'addend2'.
    addend1 = randomizer.Next(51);
    addend2 = randomizer.Next(51);

    // Convert the two randomly generated numbers
    // into strings so that they can be displayed
    // in the label controls.
    plusLeftLabel.Text = addend1.ToString();
    plusRightLabel.Text = addend2.ToString();

    // 'sum' is the name of the NumericUpDown control.
    // This step makes sure its value is zero before
    // adding any values to it.
    sum.Value = 0;
}

```

```

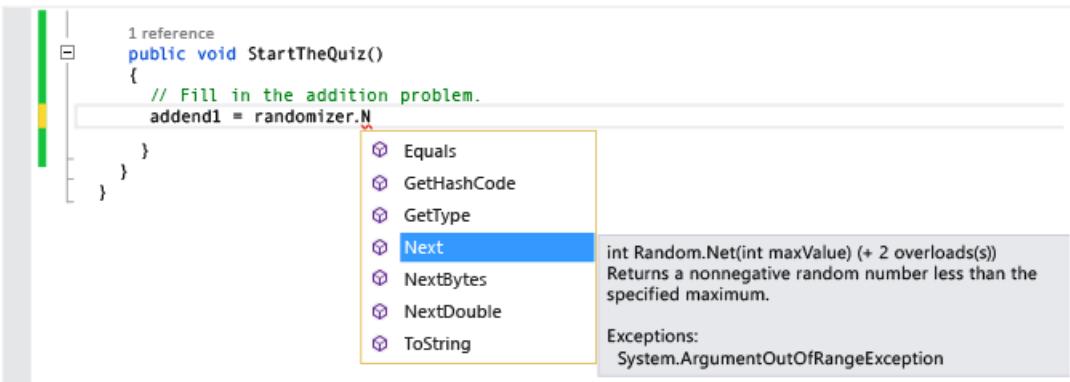
''' <summary>
''' Start the quiz by filling in all of the problems
''' and starting the timer.
''' </summary>
''' <remarks></remarks>
Public Sub StartTheQuiz()
    ' Fill in the addition problem.
    ' Generate two random numbers to add.
    ' Store the values in the variables 'addend1' and 'addend2'.
    addend1 = randomizer.Next(51)
    addend2 = randomizer.Next(51)

    ' Convert the two randomly generated numbers
    ' into strings so that they can be displayed
    ' in the label controls.
    plusLeftLabel.Text = addend1.ToString()
    plusRightLabel.Text = addend2.ToString()

    ' 'sum' is the name of the NumericUpDown control.
    ' This step makes sure its value is zero before
    ' adding any values to it.
    sum.Value = 0
End Sub

```

Notice that when you enter the dot (.) after `randomizer` in the code, an IntelliSense window opens and shows you all of the Random object's methods that you can call. For example, IntelliSense lists the `Next()` method, as follows.



Next method

When you enter a dot after an object, IntelliSense shows a list of the object's members, such as properties, methods, and events.

NOTE

When you use the `Next()` method with the `Random` object, such as when you call `randomizer.Next(50)`, you get a random number that's less than 50 (from 0 through 49). In this example, you called `randomizer.Next(51)`. You used 51 and not 50 so that the two random numbers will add up to an answer that's from 0 through 100. If you pass 50 to the `Next()` method, it chooses a number from 0 through 49, so the highest possible answer is 98, not 100. After the first two statements in the method run, each of the two integer variables, `addend1` and `addend2`, hold a random number from 0 through 50. This screenshot shows C# code, but IntelliSense works the same way for Visual Basic.

Take a closer look at these statements.

```
plusLeftLabel.Text = addend1.ToString();
plusRightLabel.Text = addend2.ToString();
```

```
' Convert the two randomly generated numbers
' into strings so that they can be displayed
' in the label controls.
plusLeftLabel.Text = addend1.ToString()
plusRightLabel.Text = addend2.ToString()
```

The statements set the **Text** properties of **plusLeftLabel** and **plusRightLabel** so that they display the two random numbers. You must use the integer's `ToString()` method to convert the numbers to text. (In programming, string means text. Label controls display only text, not numbers.)

6. In the design window, either double-click the **Start** button, or choose it and then choose the **Enter** key.

When a quiz taker chooses this button, the quiz should start, and you've just added a Click event handler to implement that behavior.

7. Add the following two statements.

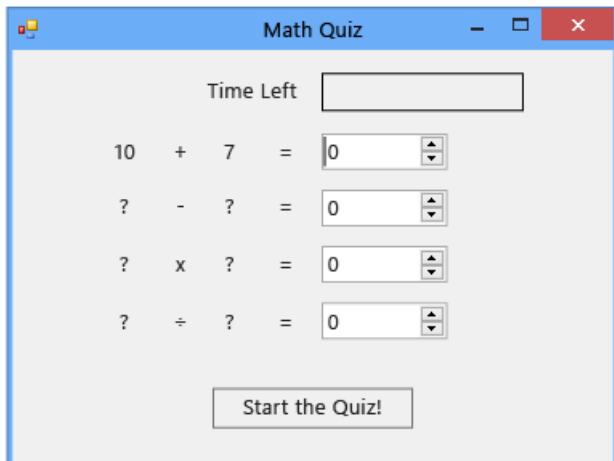
```
private void startButton_Click(object sender, EventArgs e)
{
    StartTheQuiz();
    startButton.Enabled = false;
}
```

```
' Call the StartTheQuiz() method and enable  
' the Start button.  
Private Sub startButton_Click() Handles startButton.Click  
    StartTheQuiz()  
    startButton.Enabled = False  
End Sub
```

The first statement calls the new `StartTheQuiz()` method. The second statement sets the **Enabled** property of the **startButton** control to **False** so that the quiz taker can't choose the button during a quiz.

8. Save your code, run it, and then choose the **Start** button.

A random addition problem appears, as shown in the following screenshot.



Random addition problem

In the next step of the tutorial, you'll add the sum.

To continue or review

- To go to the next tutorial step, see [Step 3: Add a countdown timer](#).
- To return to the previous tutorial step, see [Step 1: Create a project and add labels to your form](#).

Step 3: Add a countdown timer

10/15/2019 • 6 minutes to read • [Edit Online](#)

In the third part of this tutorial, you'll add a countdown timer to track the number of seconds that remain for the quiz taker to finish.

NOTE

This topic is part of a tutorial series about basic coding concepts.

- For an overview of the tutorial, see [Tutorial 2: Create a timed math quiz](#).
- To download a completed version of the code, see [Complete math quiz tutorial sample](#).

To add a countdown timer

1. Add an integer variable that's named **timeLeft**, just like you did in the previous procedure. Your code should look like the following.

```
Public Class Form1

    ' Create a Random object called randomizer
    ' to generate random numbers.
    Private randomizer As New Random

    ' These integer variables store the numbers
    ' for the addition problem.
    Private addend1 As Integer
    Private addend2 As Integer

    ' This integer variable keeps track of the
    ' remaining time.
    Private timeLeft As Integer
```

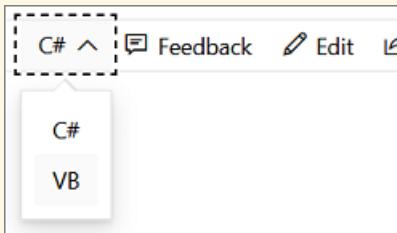
```
public partial class Form1 : Form
{
    // Create a Random object called randomizer
    // to generate random numbers.
    Random randomizer = new Random();

    // These integer variables store the numbers
    // for the addition problem.
    int addend1;
    int addend2;

    // This integer variable keeps track of the
    // remaining time.
    int timeLeft;
```

IMPORTANT

Use the programming language control at the top right of this page to view either the C# code snippet or the Visual Basic code snippet.



Now you need a method that actually counts the seconds, such as a timer, which raises an event after the amount of time that you specify.

2. In the design window, move a **Timer** control from the **Components** category of the **Toolbox** to your form.

The control appears in the gray area at the bottom of the design window.

3. On the form, choose the **timer1** icon that you just added, and set its **Interval** property to **1000**.

Because the interval value is milliseconds, a value of 1000 causes the **Tick** event to fire every second.

4. On the form, double-click the **Timer** control, or choose it and then choose the **Enter** key.

The code editor appears and displays the method for the Tick event handler that you just added.

5. Add the following statements to the new event handler method.

```
Private Sub Timer1_Tick() Handles Timer1.Tick

    If timeLeft > 0 Then
        ' Display the new time left
        ' by updating the Time Left label.
        timeLeft -= 1
        timeLabel.Text = timeLeft & " seconds"
    Else
        ' If the user ran out of time, stop the timer, show
        ' a MessageBox, and fill in the answers.
        Timer1.Stop()
        timeLabel.Text = "Time's up!"
        MessageBox.Show("You didn't finish in time.", "Sorry!")
        sum.Value = addend1 + addend2
        startButton.Enabled = True
    End If

End Sub
```

```

private void timer1_Tick(object sender, EventArgs e)
{
    if (timeLeft > 0)
    {
        // Display the new time left
        // by updating the Time Left label.
        timeLeft = timeLeft - 1;
        timeLabel.Text = timeLeft + " seconds";
    }
    else
    {
        // If the user ran out of time, stop the timer, show
        // a MessageBox, and fill in the answers.
        timer1.Stop();
        timeLabel.Text = "Time's up!";
        MessageBox.Show("You didn't finish in time.", "Sorry!");
        sum.Value = addend1 + addend2;
        startButton.Enabled = true;
    }
}

```

Based on what you added, the timer checks each second whether time has run out by determining whether the **timeLeft** integer variable is greater than 0. If it is, time still remains. The timer first subtracts 1 from timeLeft and then updates the **Text** property of the **timeLabel** control to show the quiz taker how many seconds remain.

If no time remains, the timer stops and changes the text of the **timeLabel** control so that it shows **Time's up!** A message box announces that the quiz is over, and the answer is revealed—in this case, by adding addend1 and addend2. The **Enabled** property of the **startButton** control is set to **true** so that the quiz taker can start another quiz.

You just added an **if else** statement, which is how you tell programs to make decisions. An **if else** statement looks like the following.

NOTE

The following example is for demonstration only--don't add it to your project.

```

If (something that your program will check) Then
    ' One or more statements that will run
    ' if what the program checked is true.
Else
    ' One or more statements that will run
    ' if what the program checked is false.
End If

```

```

if (something that your program will check)
{
    // One or more statements that will run
    // if what the program checked is true.
}
else
{
    // One or more statements that will run
    // if what the program checked is false.
}

```

Look closely at the statement that you added in the **else** block to show the answer to the addition

problem.

```
sum.Value = addend1 + addend2
```

```
sum.Value = addend1 + addend2;
```

The statement `addend1 + addend2` adds the values in the two variables together. The first part (`sum.Value`) uses the **Value** property of the sum NumericUpDown control to display the correct answer. You use the same property later to check the answers for the quiz.

Quiz takers can enter numbers more easily by using a [NumericUpDown](#) control, which is why you use one for the answers to the math problems. All of the potential answers are whole numbers from 0 through 100. By leaving the default values of the **Minimum**, **Maximum**, and **DecimalPlaces** properties, you ensure that quiz takers can't enter decimals, negative numbers, or numbers that are too high. (If you wanted to allow quiz takers to enter 3.141 but not 3.1415, you could set the **DecimalPlaces** property to 3.)

6. Add three lines to the end of the `StartTheQuiz()` method, so the code looks like the following.

```
''' <summary>
''' Start the quiz by filling in all of the problem
''' values and starting the timer.
''' </summary>
''' <remarks></remarks>
Public Sub StartTheQuiz()

    ' Fill in the addition problem.
    ' Generate two random numbers to add.
    ' Store the values in the variables 'addend1' and 'addend2'.
    addend1 = randomizer.Next(51)
    addend2 = randomizer.Next(51)

    ' Convert the two randomly generated numbers
    ' into strings so that they can be displayed
    ' in the label controls.
    plusLeftLabel.Text = addend1.ToString()
    plusRightLabel.Text = addend2.ToString()

    ' 'sum' is the name of the NumericUpDown control.
    ' This step makes sure its value is zero before
    ' adding any values to it.
    sum.Value = 0

    ' Start the timer.
    timeLeft = 30
    timeLabel.Text = "30 seconds"
    Timer1.Start()

End Sub
```

```

/// <summary>
/// Start the quiz by filling in all of the problem
/// values and starting the timer.
/// </summary>
public void StartTheQuiz()
{
    // Fill in the addition problem.
    // Generate two random numbers to add.
    // Store the values in the variables 'addend1' and 'addend2'.
    addend1 = randomizer.Next(51);
    addend2 = randomizer.Next(51);

    // Convert the two randomly generated numbers
    // into strings so that they can be displayed
    // in the label controls.
    plusLeftLabel.Text = addend1.ToString();
    plusRightLabel.Text = addend2.ToString();

    // 'sum' is the name of the NumericUpDown control.
    // This step makes sure its value is zero before
    // adding any values to it.
    sum.Value = 0;

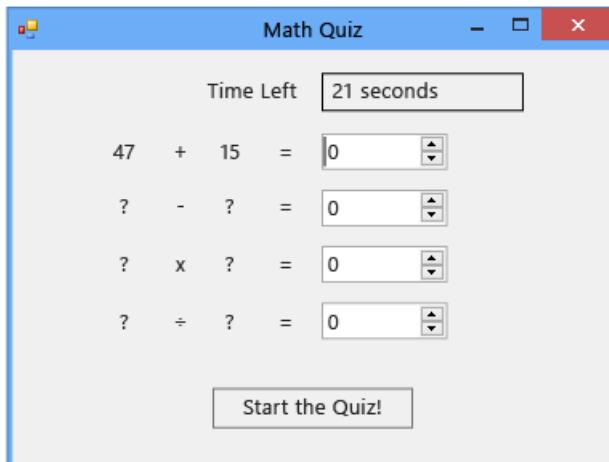
    // Start the timer.
    timeLeft = 30;
    timeLabel.Text = "30 seconds";
    timer1.Start();
}

```

Now, when your quiz starts, the **timeLeft** variable is set to 30 and the **Text** property of the **timeLabel** control is set to 30 seconds. Then the [Start\(\)](#) method of the Timer control starts the countdown. (The quiz doesn't check the answer yet—that comes next.)

7. Save your program, run it, and then choose the **Start** button on the form.

The timer starts to count down. When time runs out, the quiz ends, and the answer appears. The following illustration shows the quiz in progress.



Math quiz in progress

To continue or review

- To go to the next tutorial step, see [Step 4: Add the CheckTheAnswer\(\) method](#).
- To return to the previous tutorial step, see [Step 2: Create a random addition problem](#).

Step 4: Add the CheckTheAnswer() method

10/15/2019 • 3 minutes to read • [Edit Online](#)

In the fourth part of this tutorial, you'll write a method, `checkTheAnswer()`, that determines whether the answers to the math problems are correct. This topic is part of a tutorial series about basic coding concepts. For an overview of the tutorial, see [Tutorial 2: Create a timed math quiz](#).

NOTE

This topic is part of a tutorial series about basic coding concepts.

- For an overview of the tutorial, see [Tutorial 2: Create a timed math quiz](#).
- To download a completed version of the code, see [Complete math quiz tutorial sample](#).

To verify whether the answers are correct

NOTE

If you're following along in Visual Basic, you'll use the `Function` keyword instead of the usual `sub` keyword because this method returns a value. It's really that simple: a sub doesn't return a value, but a function does.

1. Add the `CheckTheAnswer()` method.

When this method is called, it adds the values of `addend1` and `addend2` and compares the result to the value in the sum `NumericUpDown` control. If the values are equal, the method returns a value of `true`. Otherwise, the method returns a value of `false`. Your code should look like the following.

```
''' <summary>
''' Check the answer to see if the user got everything right.
''' </summary>
''' <returns>True if the answer's correct, false otherwise.</returns>
''' <remarks></remarks>
Public Function CheckTheAnswer() As Boolean

    If addend1 + addend2 = sum.Value Then
        Return True
    Else
        Return False
    End If

End Function
```

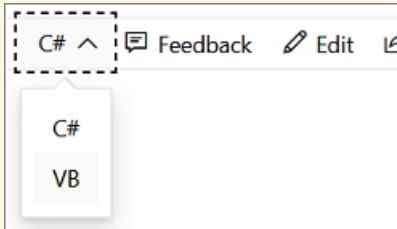
```

/// <summary>
/// Check the answer to see if the user got everything right.
/// </summary>
/// <returns>True if the answer's correct, false otherwise.</returns>
private bool CheckTheAnswer()
{
    if (addend1 + addend2 == sum.Value)
        return true;
    else
        return false;
}

```

IMPORTANT

Use the programming language control at the top right of this page to view either the C# code snippet or the Visual Basic code snippet.



Next, you'll check the answer by updating the code in the method for the timer's [Tick](#) event handler to call the new `CheckTheAnswer()` method.

2. Add the following code to the `if else` statement.

```

Private Sub Timer1_Tick() Handles Timer1.Tick

If CheckTheAnswer() Then
    ' If CheckTheAnswer() returns true, then the user
    ' got the answer right. Stop the timer
    ' and show a MessageBox.
    Timer1.Stop()
    MessageBox.Show("You got all of the answers right!", "Congratulations!")
    startButton.Enabled = True
ElseIf timeLeft > 0 Then
    ' If CheckTheAnswer() return false, keep counting
    ' down. Decrease the time left by one second and
    ' display the new time left by updating the
    ' Time Left label.
    timeLeft -= 1
    timeLabel.Text = timeLeft & " seconds"
Else
    ' If the user ran out of time, stop the timer, show
    ' a MessageBox, and fill in the answers.
    Timer1.Stop()
    timeLabel.Text = "Time's up!"
    MessageBox.Show("You didn't finish in time.", "Sorry!")
    sum.Value = addend1 + addend2
    startButton.Enabled = True
End If

End Sub

```

```

private void timer1_Tick(object sender, EventArgs e)
{
    if (CheckTheAnswer())
    {
        // If CheckTheAnswer() returns true, then the user
        // got the answer right. Stop the timer
        // and show a MessageBox.
        timer1.Stop();
        MessageBox.Show("You got all the answers right!",
                       "Congratulations!");
        startButton.Enabled = true;
    }
    else if (timeLeft > 0)
    {
        // If CheckTheAnswer() return false, keep counting
        // down. Decrease the time left by one second and
        // display the new time left by updating the
        // Time Left label.
        timeLeft--;
        timeLabel.Text = timeLeft + " seconds";
    }
    else
    {
        // If the user ran out of time, stop the timer, show
        // a MessageBox, and fill in the answers.
        timer1.Stop();
        timeLabel.Text = "Time's up!";
        MessageBox.Show("You didn't finish in time.", "Sorry!");
        sum.Value = addend1 + addend2;
        startButton.Enabled = true;
    }
}

```

If the answer is correct, `CheckTheAnswer()` returns `true`. The event handler stops the timer, shows a congratulatory message, and then makes the **Start** button available again. Otherwise, the quiz continues.

- Save your program, run it, start a quiz, and provide a correct answer to the addition problem.

NOTE

When you enter your answer, you must either select the default value before you start to enter your answer, or you must delete the zero manually. You'll correct this behavior later in this tutorial.

When you provide a correct answer, a message box opens, the **Start** button becomes available, and the timer stops.

To continue or review

- To go to the next tutorial step, see [Step 5: Add Enter event handlers for the NumericUpDown controls.](#)
- To return to the previous tutorial step, see [Step 3: Add a countdown timer.](#)

Step 5: Add Enter event handlers for the NumericUpDown controls

10/15/2019 • 4 minutes to read • [Edit Online](#)

In the fifth part of this tutorial, you'll add **Enter** event handlers to make entering answers for quiz problems a little easier. This code will select and clear the current value in each **NumericUpDown** control as soon as the quiz taker chooses it and starts to enter a different value.

NOTE

This topic is part of a tutorial series about basic coding concepts.

- For an overview of the tutorial, see [Tutorial 2: Create a timed math quiz](#).
- To download a completed version of the code, see [Complete math quiz tutorial sample](#).

To verify the default behavior

1. Run your program, and start the quiz.

In the **NumericUpDown** control for the addition problem, the cursor flashes next to **0** (zero).

2. Enter **3**, and note that the control shows **30**.
3. Enter **5**, and note that **350** appears but changes to **100** after a second.

Before you fix this problem, think about what's happening. Consider why the **0** didn't disappear when you entered **3** and why **350** changed to **100** but not immediately.

This behavior may seem odd, but it makes sense given the logic of the code. When you choose the **Start** button, its **Enabled** property is set to **False**, and the button appears dimmed and is unavailable. Your program changes the current selection (focus) to the control that has the next lowest TabIndex value, which is the **NumericUpDown** control for the addition problem. When you use the **Tab** key to go to a **NumericUpDown** control, the cursor is automatically positioned at the start of the control, which is why the numbers that you enter appear from the left side and not the right side. When you specify a number that's higher than the value of the **MaximumValue** property, which is set to 100, the number that you enter is replaced with the value of that property.

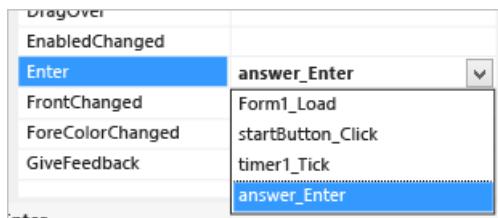
To add an Enter event handler for a **NumericUpDown** control

1. Choose the first **NumericUpDown** control (named "sum") on the form, and then, in the **Properties** dialog box, choose the **Events** icon on the toolbar.



The **Events** tab in the **Properties** dialog box displays all of the events that you can respond to (handle) for the item that you choose on the form. Because you chose the **NumericUpDown** control, all of the events listed pertain to it.

2. Choose the **Enter** event, type `answer_Enter`, and then press the **Enter** key.



You've just added an Enter event handler for the sum NumericUpDown control, and you've named the handler **answer_Enter**.

3. In the method for the **answer_Enter** event handler, add the following code:

```
''' <summary>
''' Modify the behavior of the NumericUpDown control
''' to make it easier to enter numeric values for
''' the quiz.
''' </summary>
Private Sub answer_Enter(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles sum.Enter

    ' Select the whole answer in the NumericUpDown control.
    Dim answerBox = TryCast(sender, NumericUpDown)

    If answerBox IsNot Nothing Then
        Dim lengthOfAnswer = answerBox.Value.ToString().Length
        answerBox.Select(0, lengthOfAnswer)
    End If

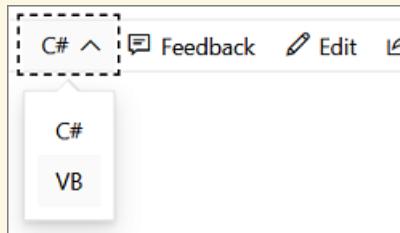
End Sub
```

```
private void answer_Enter(object sender, EventArgs e)
{
    // Select the whole answer in the NumericUpDown control.
    NumericUpDown answerBox = sender as NumericUpDown;

    if (answerBox != null)
    {
        int lengthOfAnswer = answerBox.Value.ToString().Length;
        answerBox.Select(0, lengthOfAnswer);
    }
}
```

IMPORTANT

Use the programming language control at the top right of this page to view either the C# code snippet or the Visual Basic code snippet.



This code may look complex, but you can understand it if you look at it step by step. First, look at the top of the method: `object sender` in C# or `sender As System.Object` in Visual Basic. This parameter refers to the object whose event is firing, which is known as the sender. In this case, the sender object is the NumericUpDown control. So, in the first line of the method, you specify that the sender isn't just any generic object but specifically a NumericUpDown control. (Every NumericUpDown control is an object, but

not every object is a NumericUpDown control.) The NumericUpDown control is named **answerBox** in this method, because it will be used for all of the NumericUpDown controls on the form, not just the sum NumericUpDown control. Because you declare the **answerBox** variable in this method, its scope applies only to this method. In other words, the variable can be used only within this method.

The next line verifies whether **answerBox** was successfully converted (cast) from an object to a NumericUpDown control. If the conversion was unsuccessful, the variable would have a value of `null` (C#) or `Nothing` (Visual Basic). The third line gets the length of the answer that appears in the NumericUpDown control, and the fourth line selects the current value in the control based on this length. Now, when the quiz taker chooses the control, Visual Studio fires this event, which causes the current answer to be selected. As soon as the quiz taker starts to enter a different answer, the previous answer is cleared and replaced with the new answer.

4. In **Windows Forms Designer**, choose the difference **NumericUpDown** control.
5. In the **Events** page of the **Properties** dialog box, scroll down to the **Enter** event, choose the drop-down arrow at the end of the row, and then choose the `answer_Enter` event handler that you just added.
6. Repeat the previous step for the product and quotient NumericUpDown controls.
7. Save your program, and then run it.

When you choose a **NumericUpDown** control, the existing value is automatically selected and then cleared when you start to enter a different value.

To continue or review

- To go to the next tutorial step, see [Step 6: Add a subtraction problem](#).
- To return to the previous tutorial step, see [Step 4: Add the CheckTheAnswer\(\) method](#).

Step 6: Add a subtraction problem

10/17/2019 • 6 minutes to read • [Edit Online](#)

In the sixth part of this tutorial, you'll add a subtraction problem and learn how to perform the following tasks:

- Store the subtraction values.
- Generate random numbers for the problem (and be sure that the answer is between 0 and 100).
- Update the method that checks the answers so that it checks the new subtraction problem too.
- Update your timer's [Tick](#) event handler so that the event handler fills in the correct answer when time runs out.

NOTE

This topic is part of a tutorial series about basic coding concepts.

- For an overview of the tutorial, see [Tutorial 2: Create a timed math quiz](#).
- To download a completed version of the code, see [Complete math quiz tutorial sample](#).

To add a subtraction problem

1. Add two integer variables for the subtraction problem to your form, between the integer variables for the addition problem and the timer. The code should look like the following.

```
Public Class Form1

    ' Create a Random object called randomizer
    ' to generate random numbers.
    Private randomizer As New Random

    ' These integer variables store the numbers
    ' for the addition problem.
    Private addend1 As Integer
    Private addend2 As Integer

    ' These integer variables store the numbers
    ' for the subtraction problem.
    Private minuend As Integer
    Private subtrahend As Integer

    ' This integer variable keeps track of the
    ' remaining time.
    Private timeLeft As Integer
```

```
public partial class Form1 : Form
{
    // Create a Random object called randomizer
    // to generate random numbers.
    Random randomizer = new Random();

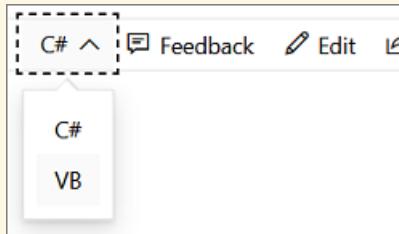
    // These integer variables store the numbers
    // for the addition problem.
    int addend1;
    int addend2;

    // These integer variables store the numbers
    // for the subtraction problem.
    int minuend;
    int subtrahend;

    // This integer variable keeps track of the
    // remaining time.
    int timeLeft;
```

IMPORTANT

Use the programming language control at the top right of this page to view either the C# code snippet or the Visual Basic code snippet.



The names of the new integer variables—**minuend** and **subtrahend**—aren't programming terms. They're the traditional names in arithmetic for the number that's being subtracted (the subtrahend) and the number from which the subtrahend is being subtracted (the minuend). The difference is the minuend minus the subtrahend. You could use other names, because your program doesn't require specific names for variables, controls, components, or methods. You must follow rules such as not starting names with digits, but you can generally use names such as x1, x2, x3, and x4. However, generic names make code difficult to read and problems nearly impossible to track down. To keep variable names unique and helpful, you'll use the traditional names for multiplication (multiplicand × multiplier = product) and division (dividend ÷ divisor = quotient) later in this tutorial.

Next, you'll modify the `StartTheQuiz()` method to provide random values for the subtraction problem.

2. Add the following code after the "Fill in the subtraction problem" comment.

```
''' <summary>
''' Start the quiz by filling in all of the problem
''' values and starting the timer.
''' </summary>
''' <remarks></remarks>
Public Sub StartTheQuiz()

    ' Fill in the addition problem.
    ' Generate two random numbers to add.
    ' Store the values in the variables 'addend1' and 'addend2'.
    addend1 = randomizer.Next(51)
    addend2 = randomizer.Next(51)

    ' Convert the two randomly generated numbers
    ' into strings so that they can be displayed
    ' in the label controls.
    plusLeftLabel.Text = addend1.ToString()
    plusRightLabel.Text = addend2.ToString()

    ' 'sum' is the name of the NumericUpDown control.
    ' This step makes sure its value is zero before
    ' adding any values to it.
    sum.Value = 0

    ' Fill in the subtraction problem.
    minuend = randomizer.Next(1, 101)
    subtrahend = randomizer.Next(1, minuend)
    minusLeftLabel.Text = minuend.ToString()
    minusRightLabel.Text = subtrahend.ToString()
    difference.Value = 0

    ' Start the timer.
    timeLeft = 30
    timeLabel.Text = "30 seconds"
    Timer1.Start()

End Sub
```

```

/// <summary>
/// Start the quiz by filling in all of the problem
/// values and starting the timer.
/// </summary>
public void StartTheQuiz()
{
    // Fill in the addition problem.
    // Generate two random numbers to add.
    // Store the values in the variables 'addend1' and 'addend2'.
    addend1 = randomizer.Next(51);
    addend2 = randomizer.Next(51);

    // Convert the two randomly generated numbers
    // into strings so that they can be displayed
    // in the label controls.
    plusLeftLabel.Text = addend1.ToString();
    plusRightLabel.Text = addend2.ToString();

    // 'sum' is the name of the NumericUpDown control.
    // This step makes sure its value is zero before
    // adding any values to it.
    sum.Value = 0;

    // Fill in the subtraction problem.
    minuend = randomizer.Next(1, 101);
    subtrahend = randomizer.Next(1, minuend);
    minusLeftLabel.Text = minuend.ToString();
    minusRightLabel.Text = subtrahend.ToString();
    difference.Value = 0;

    // Start the timer.
    timeLeft = 30;
    timeLabel.Text = "30 seconds";
    timer1.Start();
}

```

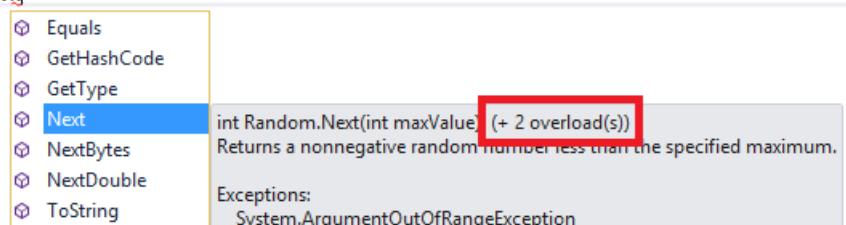
To prevent negative answers for the subtraction problem, this code uses the `Next()` method of the `Random` class a little differently from how the addition problem does. When you give the `Next()` method two values, it picks a random number that's greater than or equal to the first value and less than the second one. The following code chooses a random number from 1 through 100 and stores it in the `minuend` variable.

```
minuend = randomizer.Next(1, 101)
```

```
minuend = randomizer.Next(1, 101);
```

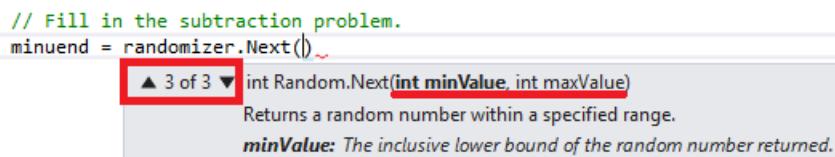
You can call the `Next()` method of the `Random` class, which you named "randomizer" earlier in this tutorial, in multiple ways. Methods that you can call in more than one way are referred to as overloaded, and you can use IntelliSense to explore them. Look again at the tooltip of the IntelliSense window for the `Next()` method.

```
// Fill in the subtraction problem.
minuend = randomizer.N
```



IntelliSense window tooltip

The tooltip shows **(+ 2 overload(s))**, which means that you can call the `Next()` method in two other ways. Overloads contain different numbers or types of arguments, so that they work slightly differently from one another. For example, a method might take a single integer argument, and one of its overloads might take an integer and a string. You choose the correct overload based on what you want it to do. When you add the code to the `StartTheQuiz()` method, more information appears in the IntelliSense window as soon as you enter `randomizer.Next()`. To cycle through the overloads, choose the **Up Arrow** and **Down Arrow** keys as shown in the following illustration:



Overload for `Next()` method in IntelliSense

In this case, you want to choose the last overload, because you can specify minimum and maximum values.

3. Modify the `CheckTheAnswer()` method to check for the correct subtraction answer.

```
''' <summary>
''' Check the answers to see if the user got everything right.
''' </summary>
''' <returns>True if the answer's correct, false otherwise.</returns>
''' <remarks></remarks>
Public Function CheckTheAnswer() As Boolean

    If addend1 + addend2 = sum.Value AndAlso
        minuend - subtrahend = difference.Value Then

        Return True
    Else
        Return False
    End If

End Function
```

```
/// <summary>
/// Check the answers to see if the user got everything right.
/// </summary>
/// <returns>True if the answer's correct, false otherwise.</returns>
private bool CheckTheAnswer()
{
    if ((addend1 + addend2 == sum.Value)
        && (minuend - subtrahend == difference.Value))
        return true;
    else
        return false;
}
```

In C#, `&&` is the `logical and` operator. In Visual Basic, the equivalent operator is `AndAlso`. These operators indicate "If the sum of `addend1` and `addend2` equals the value of the `sum` `NumericUpDown` and if `minuend` minus `subtrahend` equals the value of the `difference` `NumericUpDown`." The `CheckTheAnswer()` method returns `true` only if the answers to the addition and the subtraction problems are both correct.

4. Replace the last part of the timer's Tick event handler with the following code so that it fills in the correct answer when time runs out.

```

Else
    ' If the user ran out of time, stop the timer, show
    ' a MessageBox, and fill in the answers.
    Timer1.Stop()
    timeLabel.Text = "Time's up!"
    MessageBox.Show("You didn't finish in time.", "Sorry!")
    sum.Value = addend1 + addend2
    difference.Value = minuend - subtrahend
    startButton.Enabled = True
End If

```

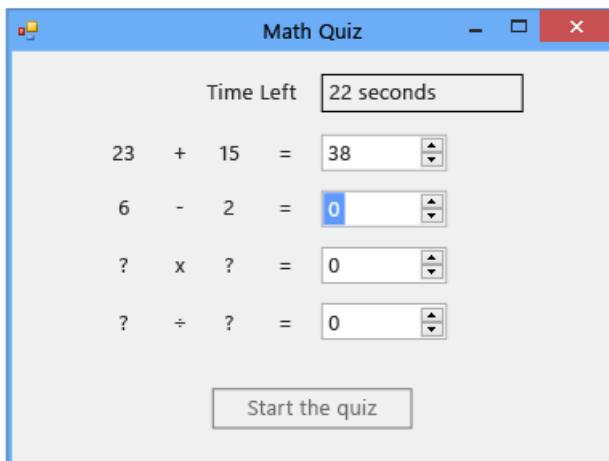
```

else
{
    // If the user ran out of time, stop the timer, show
    // a MessageBox, and fill in the answers.
    timer1.Stop();
    timeLabel.Text = "Time's up!";
    MessageBox.Show("You didn't finish in time.", "Sorry!");
    sum.Value = addend1 + addend2;
    difference.Value = minuend - subtrahend;
    startButton.Enabled = true;
}

```

5. Save and run your code.

Your program includes a subtraction problem, as the following illustration shows:



Math quiz with subtraction problem

To continue or review

- To go to the next tutorial step, see [Step 7: Add multiplication and division problems](#).
- To return to the previous tutorial step, see [Step 5: Add Enter event handlers for the NumericUpDown controls](#).

Step 7: Add multiplication and division problems

10/18/2019 • 5 minutes to read • [Edit Online](#)

In the seventh part of this tutorial, you'll add multiplication and division problems, but first think about how to make that change. Consider the initial step, which involves storing values.

NOTE

This topic is part of a tutorial series about basic coding concepts.

- For an overview of the tutorial, see [Tutorial 2: Create a timed math quiz](#).
- To download a completed version of the code, see [Complete math quiz tutorial sample](#).

To add multiplication and division problems

1. Add four more integer variables to the form.

```
Public Class Form1

    ' Create a Random object called randomizer
    ' to generate random numbers.
    Private randomizer As New Random

    ' These integer variables store the numbers
    ' for the addition problem.
    Private addend1 As Integer
    Private addend2 As Integer

    ' These integer variables store the numbers
    ' for the subtraction problem.
    Private minuend As Integer
    Private subtrahend As Integer

    ' These integer variables store the numbers
    ' for the multiplication problem.
    Private multiplicand As Integer
    Private multiplier As Integer

    ' These integer variables store the numbers
    ' for the division problem.
    Private dividend As Integer
    Private divisor As Integer

    ' This integer variable keeps track of the
    ' remaining time.
    Private timeLeft As Integer
```

```
public partial class Form1 : Form
{
    // Create a Random object called randomizer
    // to generate random numbers.
    Random randomizer = new Random();

    // These integer variables store the numbers
    // for the addition problem.
    int addend1;
    int addend2;

    // These integer variables store the numbers
    // for the subtraction problem.
    int minuend;
    int subtrahend;

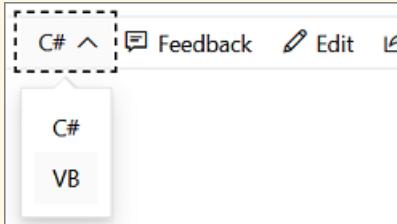
    // These integer variables store the numbers
    // for the multiplication problem.
    int multiplicand;
    int multiplier;

    // These integer variables store the numbers
    // for the division problem.
    int dividend;
    int divisor;

    // This integer variable keeps track of the
    // remaining time.
    int timeLeft;
```

IMPORTANT

Use the programming language control at the top right of this page to view either the C# code snippet or the Visual Basic code snippet.



2. As you did before, modify the `StartTheQuiz()` method to fill in random numbers for the multiplication and division problems.

```

''' <summary>
''' Start the quiz by filling in all of the problem
''' values and starting the timer.
''' </summary>
''' <remarks></remarks>
Public Sub StartTheQuiz()

    ' Fill in the addition problem.
    ' Generate two random numbers to add.
    ' Store the values in the variables 'addend1' and 'addend2'.
    addend1 = randomizer.Next(51)
    addend2 = randomizer.Next(51)

    ' Convert the two randomly generated numbers
    ' into strings so that they can be displayed
    ' in the label controls.
    plusLeftLabel.Text = addend1.ToString()
    plusRightLabel.Text = addend2.ToString()

    ' 'sum' is the name of the NumericUpDown control.
    ' This step makes sure its value is zero before
    ' adding any values to it.
    sum.Value = 0

    ' Fill in the subtraction problem.
    minuend = randomizer.Next(1, 101)
    subtrahend = randomizer.Next(1, minuend)
    minusLeftLabel.Text = minuend.ToString()
    minusRightLabel.Text = subtrahend.ToString()
    difference.Value = 0

    ' Fill in the multiplication problem.
    multiplicand = randomizer.Next(2, 11)
    multiplier = randomizer.Next(2, 11)
    timesLeftLabel.Text = multiplicand.ToString()
    timesRightLabel.Text = multiplier.ToString()
    product.Value = 0

    ' Fill in the division problem.
    divisor = randomizer.Next(2, 11)
    Dim temporaryQuotient As Integer = randomizer.Next(2, 11)
    dividend = divisor * temporaryQuotient
    dividedLeftLabel.Text = dividend.ToString()
    dividedRightLabel.Text = divisor.ToString()
    quotient.Value = 0

    ' Start the timer.
    timeLeft = 30
    timeLabel.Text = "30 seconds"
    Timer1.Start()

End Sub

```

```

/// <summary>
/// Start the quiz by filling in all of the problem
/// values and starting the timer.
/// </summary>
public void StartTheQuiz()
{
    // Fill in the addition problem.
    // Generate two random numbers to add.
    // Store the values in the variables 'addend1' and 'addend2'.
    addend1 = randomizer.Next(51);
    addend2 = randomizer.Next(51);

    // Convert the two randomly generated numbers
    // into strings so that they can be displayed
    // in the label controls.
    plusLeftLabel.Text = addend1.ToString();
    plusRightLabel.Text = addend2.ToString();

    // 'sum' is the name of the NumericUpDown control.
    // This step makes sure its value is zero before
    // adding any values to it.
    sum.Value = 0;

    // Fill in the subtraction problem.
    minuend = randomizer.Next(1, 101);
    subtrahend = randomizer.Next(1, minuend);
    minusLeftLabel.Text = minuend.ToString();
    minusRightLabel.Text = subtrahend.ToString();
    difference.Value = 0;

    // Fill in the multiplication problem.
    multiplicand = randomizer.Next(2, 11);
    multiplier = randomizer.Next(2, 11);
    timesLeftLabel.Text = multiplicand.ToString();
    timesRightLabel.Text = multiplier.ToString();
    product.Value = 0;

    // Fill in the division problem.
    divisor = randomizer.Next(2, 11);
    int temporaryQuotient = randomizer.Next(2, 11);
    dividend = divisor * temporaryQuotient;
    dividedLeftLabel.Text = dividend.ToString();
    dividedRightLabel.Text = divisor.ToString();
    quotient.Value = 0;

    // Start the timer.
    timeLeft = 30;
    timeLabel.Text = "30 seconds";
    timer1.Start();
}

```

3. Modify the `CheckTheAnswer()` method so that it also checks the multiplication and division problems.

```

''' <summary>
''' Check the answers to see if the user got everything right.
''' </summary>
''' <returns>True if the answer's correct, false otherwise.</returns>
''' <remarks></remarks>
Public Function CheckTheAnswer() As Boolean

If addend1 + addend2 = sum.Value AndAlso
    minuend - subtrahend = difference.Value AndAlso
    multiplicand * multiplier = product.Value AndAlso
    dividend / divisor = quotient.Value Then

    Return True
Else
    Return False
End If

End Function

```

```

/// <summary>
/// Check the answers to see if the user got everything right.
/// </summary>
/// <returns>True if the answer's correct, false otherwise.</returns>
private bool CheckTheAnswer()
{
    if ((addend1 + addend2 == sum.Value)
        && (minuend - subtrahend == difference.Value)
        && (multiplicand * multiplier == product.Value)
        && (dividend / divisor == quotient.Value))
        return true;
    else
        return false;
}

```

You can't easily enter the multiplication sign (\times) and the division sign (\div) using the keyboard, so C# and Visual Basic accept an asterisk (*) for multiplication and a slash mark (/) for division.

4. Change the last part of the timer's [Tick](#) event handler so that it fills in the correct answer when time runs out.

```

Else
    ' If the user ran out of time, stop the timer, show
    ' a MessageBox, and fill in the answers.
    Timer1.Stop()
    timeLabel.Text = "Time's up!"
    MessageBox.Show("You didn't finish in time.", "Sorry!")
    sum.Value = addend1 + addend2
    difference.Value = minuend - subtrahend
    product.Value = multiplicand * multiplier
    quotient.Value = dividend / divisor
    startButton.Enabled = True
End If

```

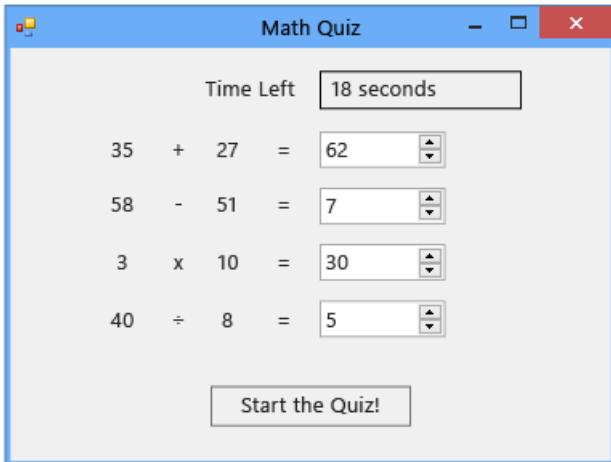
```

else
{
    // If the user ran out of time, stop the timer, show
    // a MessageBox, and fill in the answers.
    timer1.Stop();
    timeLabel.Text = "Time's up!";
    MessageBox.Show("You didn't finish in time.", "Sorry");
    sum.Value = addend1 + addend2;
    difference.Value = minuend - subtrahend;
    product.Value = multiplicand * multiplier;
    quotient.Value = dividend / divisor;
    startButton.Enabled = true;
}

```

5. Save and run your program.

Quiz takers must answer four problems to complete the quiz, as the following illustration shows.



Math quiz with four problems

To continue or review

- To go to the next tutorial step, see [Step 8: Customize the quiz](#).
- To return to the previous tutorial step, see [Step 6: Add a subtraction problem](#).

Step 8: Customize the quiz

10/15/2019 • 2 minutes to read • [Edit Online](#)

In the last part of the tutorial, you'll explore some ways to customize the quiz and expand on what you've already learned. For example, think about how the program creates random division problems for which the answer is never a fraction. To learn more, turn the `timeLabel` control a different color, and give the quiz taker a hint.

NOTE

This topic is part of a tutorial series about basic coding concepts.

- For an overview of the tutorial, see [Tutorial 2: Create a timed math quiz](#).
- To download a completed version of the code, see [Complete math quiz tutorial sample](#).

To customize the quiz

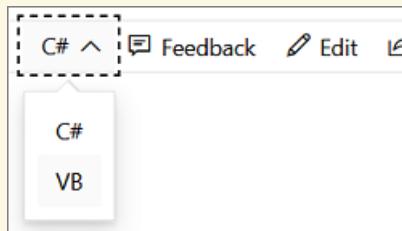
- When only five seconds remain in a quiz, turn the `timeLabel` control red by setting its **BackColor** property.

```
timeLabel.BackColor = Color.Red;
```

```
timeLabel.BackColor = Color.Red
```

IMPORTANT

Use the programming language control at the top right of this page to view either the C# code snippet or the Visual Basic code snippet.



Reset the color when the quiz is over.

- Give the quiz taker a hint by playing a sound when the correct answer is entered into a `NumericUpDown` control. (You must write an event handler for each control's `ValueChanged` event, which fires whenever the quiz taker changes the control's value.)

To continue or review

- To go to the next tutorial, see [Tutorial 3: Create a matching game](#).
- To return to the previous tutorial step, see [Step 7: Add multiplication and division problems](#).

Tutorial 3: Create a matching game

10/16/2019 • 2 minutes to read • [Edit Online](#)

In this tutorial, you build a matching game, where the player must match pairs of hidden icons.

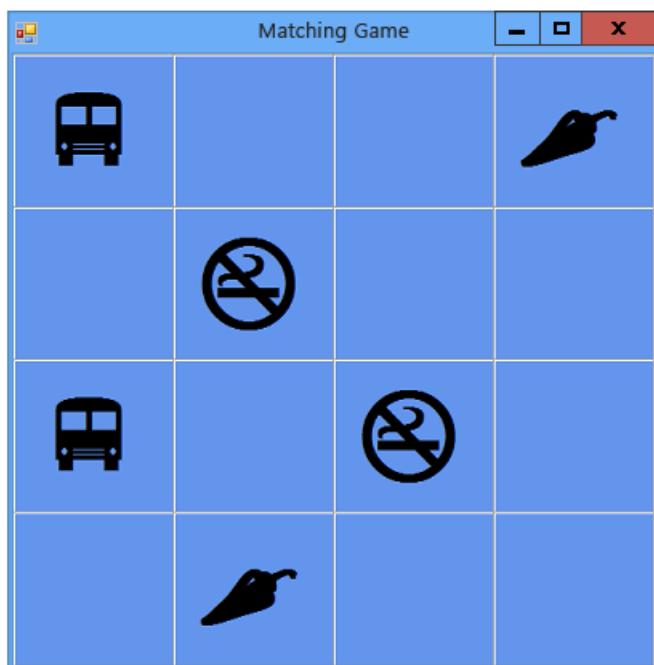
NOTE

This tutorial covers both C# and Visual Basic, so focus on the information that's specific to the programming language that you're using.

This tutorial walks you through the following tasks:

- Store objects, such as icons, in a `List<T>` object.
- Use a `foreach` loop in C# or a `For Each` loop in Visual Basic to iterate through items in a list.
- Keep track of a form's state by using reference variables.
- Build an event handler to respond to events that you can use with multiple objects.
- Make a timer that counts down and then fires an event exactly once after being started.

When you finish, your app should look similar to the following image:



Tutorial links

TITLE	DESCRIPTION
Step 1: Create a project and add a table to your form	Begin by creating the project and adding a <code>TableLayoutPanel</code> control to keep the controls aligned properly.

TITLE	DESCRIPTION
Step 2: Add a random object and a list of icons	Add a <code>Random</code> object and a <code>List</code> object, to create a list of icons.
Step 3: Assign a random icon to each label	Assign the icons randomly to the <code>Label</code> controls, so that each game is different.
Step 4: Add a click event handler to each label	Add a <code>click</code> event handler that changes the color of the label that is clicked.
Step 5: Add label references	Add reference variables to keep track of which labels are clicked.
Step 6: Add a timer	Add a timer to the form to keep track of the time that has passed in the game.
Step 7: Keep pairs visible	Keep pairs of icons visible, if a matching pair is selected.
Step 8: Add a method to verify whether the player won	Add a <code>CheckForWinner()</code> method to verify whether the player won.
Step 9: Try other features	Try other features, such as changing icons and colors, adding a grid, and adding sounds. Try making the board bigger and adjusting the timer.

There are also great, free video learning resources available to you. To learn more about programming in C#, see [C# fundamentals: Development for absolute beginners](#). To learn more about programming in Visual Basic, see [Visual Basic fundamentals: Development for absolute beginners](#).

Next steps

To begin the tutorial, start with [Step 1: Create a project and add a table to your form](#).

See also

- [More C# tutorials](#)
- [Visual Basic tutorials](#)
- [C++ tutorials](#)

Step 1: Create a project and add a table to your form

10/16/2019 • 5 minutes to read • [Edit Online](#)

The first step in creating a matching game is to create the project and add a table to your form. The table helps align the icons into an orderly 4x4 grid. You also set several properties to enhance the appearance of the game board.

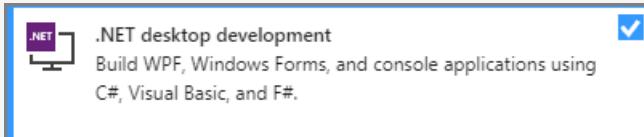
To create a project and add a table to your form

1. On the menu bar, choose **File > New > Project**.
2. Choose either **Visual C#** or **Visual Basic** on the left side of the **New Project** dialog box, and then choose **Windows Desktop**.
3. In the list of templates, choose the **Windows Forms App (.NET Framework)** template, name it *MatchingGame*, and then choose the **OK** button.

A form that's named *Form1.cs* or *Form1.vb* appears, depending on the programming language that you chose.

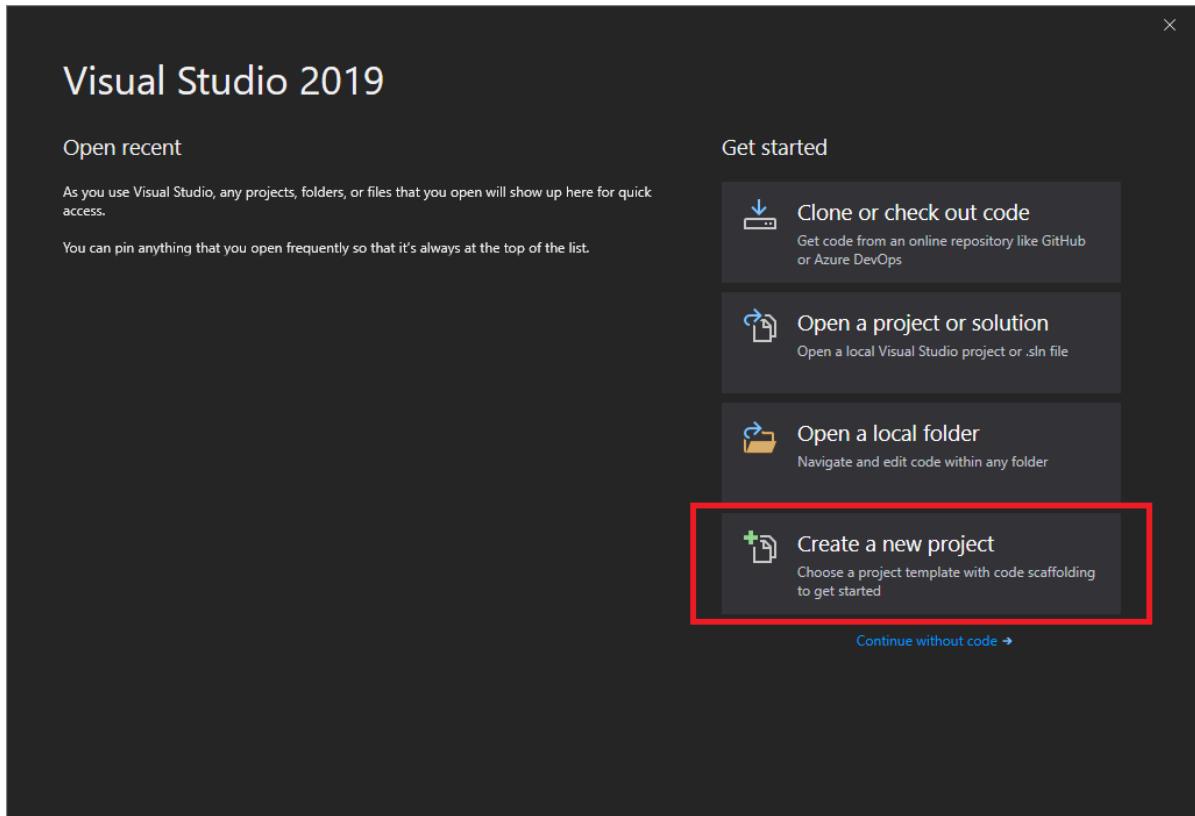
NOTE

If you don't see the **Windows Forms App (.NET Framework)** template, use the Visual Studio Installer to install the **.NET desktop development** workload.



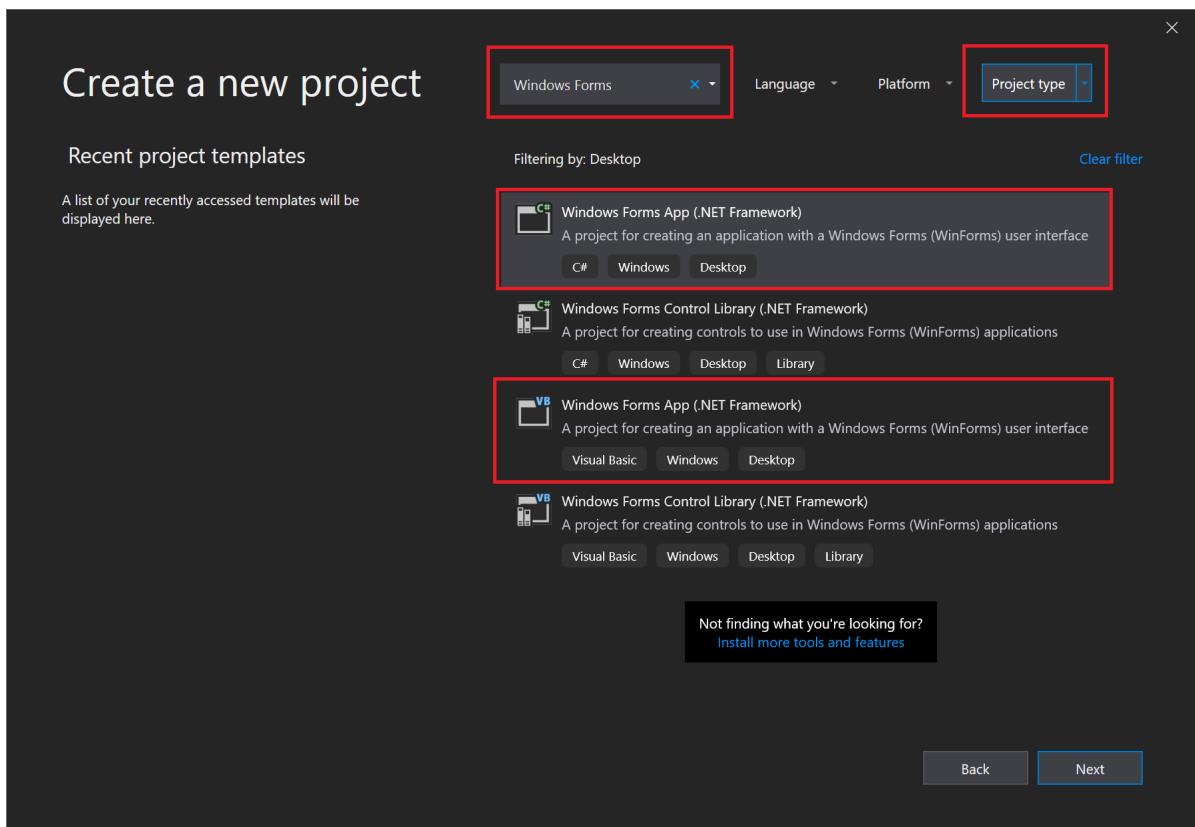
For more information, see the [Install Visual Studio](#) page.

1. On the start window, choose **Create a new project**.



2. On the **Create a new project** window, enter or type *Windows Forms* in the search box. Next, choose **Desktop** from the **Project type** list.

After you apply the **Project type** filter, choose the **Windows Forms App (.NET Framework)** template for either C# or Visual Basic, and then choose **Next**.

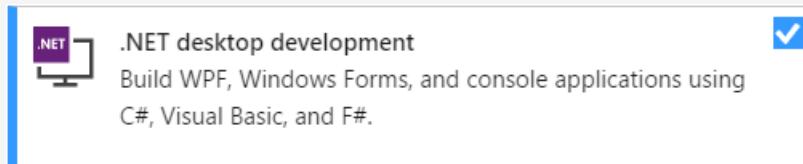


NOTE

If you do not see the **Windows Forms App (.NET Framework)** template, you can install it from the **Create a new project** window. In the **Not finding what you're looking for?** message, choose the **Install more tools and features** link.

Not finding what you're looking for?
[Install more tools and features](#)

Next, in the Visual Studio Installer, choose the Choose the **.NET desktop development** workload.



After that, choose the **Modify** button in the Visual Studio Installer. You might be prompted to save your work; if so, do so. Next, choose **Continue** to install the workload.

3. In the **Configure your new project** window, type or enter *MatchingGame* in the **Project name** box. Then, choose **Create**.

To set properties for a form

1. In the **Properties** window, set the following form properties.
 - a. Change the form's **Text** property from **Form1** to **Matching Game**. This text appears at the top of the game window.
 - b. Set the size of the form to 550 pixels wide by 550 tall. You can do this either by setting the **Size** property to **550, 550**, or by dragging the corner of the form until you see the correct size in the lower-right corner of the integrated development environment (IDE).
2. Display the toolbox by choosing the **Toolbox** tab on the left side of the IDE.
3. Drag a **TableLayoutPanel** control from the **Containers** category in the toolbox, and then set the following properties for it.
 - a. Set the **BackColor** property to **CornflowerBlue**. To do this, open the **BackColor** dialog box by choosing the drop-down arrow next to the **BackColor** property in the **Properties** window. Then, choose the **Web** tab in the **BackColor** dialog box to view a list of available color names.

NOTE

The colors are not in alphabetical order, and **CornflowerBlue** is near the bottom of the list.

- b. Set the **Dock** property to **Fill** by choosing the drop-down button next to the property and choosing the large middle button. This spreads the table out so that it covers the entire form.
- c. Set the **CellBorderStyle** property to **Inset**. This provides visual borders between each cell on the board.
- d. Choose the triangle button in the upper-right corner of the TableLayoutPanel to display its task menu.
- e. On the task menu, choose **Add Row** twice to add two more rows, and then choose **Add Column** twice to add two more columns.

- f. On the task menu, choose **Edit Rows and Columns** to open the **Column and Row Styles** window. Choose each of the columns, choose the **Percent** option button, and then set each column's width to 25 percent of the total width. Then select **Rows** from the drop-down box at the top of the window, and set each row's height to 25 percent. When you're done, choose the **OK** button.

Your TableLayoutPanel should now be a 4x4 grid, with sixteen equally sized square cells. These rows and columns are where the icon images will appear later.

4. Be certain that the TableLayoutPanel is selected in the form editor. To verify this, you should see **tableLayoutPanel1** at the top of the **Properties** window. If it is not selected, choose the TableLayoutPanel on the form, or choose it in the dropdown control at the top of the **Properties** window.

While the TableLayoutPanel is selected, open the toolbox and add a **Label** control (located in the **Common Controls** category) to the upper-left cell of the TableLayoutPanel. The label control should now be selected in the IDE. Set the following properties for it.

- a. Be sure that the label's **BackColor** property is set to **CornflowerBlue**.
- b. Set the **AutoSize** property to **False**.
- c. Set the **Dock** property to **Fill**.
- d. Set the **TextAlign** property to **MiddleCenter** by choosing the drop-down button next to the property, and then choosing the middle button. This ensures the icon appears in the middle of the cell.
- e. Choose the **Font** property. An ellipsis (...) button should appear.
- f. Choose the ellipsis button, and set the **Font** value to **Webdings**, the **Font Style** to **Bold**, and the **Size** to **48**.
- g. Set the **Text** property of the label to the letter **c**.

The upper-left cell in the TableLayoutPanel should now contain a black box centered on a blue background.

NOTE

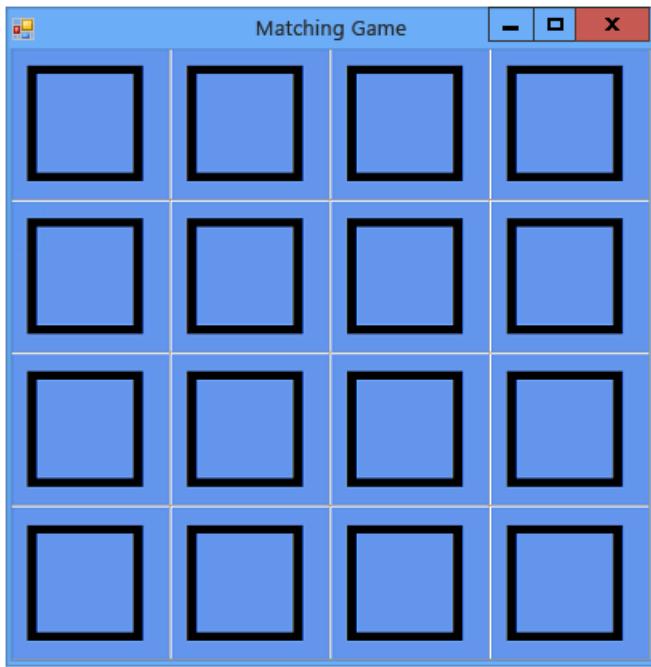
The Webdings font is a font of icons that ships with the Windows operating system. In your matching game, the player needs to match pairs of icons, so you use this font to display the icons to match. Instead of putting **c** in the **Text** property, try entering different letters to see what icons are displayed. An exclamation point is a spider, an uppercase N is an eye, and a comma is a chili pepper.

5. Choose your Label control and copy it to the next cell in the TableLayoutPanel. (Choose the **Ctrl+C** keys, or on the menu bar, choose **Edit > Copy**.) Then paste it. (Choose the **Ctrl+V** keys, or on the menu bar, choose **Edit > Paste**.) A copy of the first Label appears in the second cell of the TableLayoutPanel. Paste it again, and another Label appears in the third cell. Keep pasting Label controls until all of the cells are filled.

NOTE

If you paste too many times, the IDE adds a new row to the TableLayoutPanel so that it has a place to add your new Label control. You can undo it. To remove the new cell, choose the **Ctrl+Z** keys, or on the menu bar, choose **Edit > Undo**.

Now your form is laid out. It should look similar to the following picture.



Initial matching game form

To continue or review

- To go to the next tutorial step, see [Step 2: Add a Random object and a list of icons](#).
- To return to the overview topic, see [Tutorial 3: Create a matching game](#).

Step 2: Add a Random object and a list of icons

10/17/2019 • 4 minutes to read • [Edit Online](#)

In this step, you create a set of matching symbols for the game. Each symbol is added to two random cells in the TableLayoutPanel on the form. To do this, you use two `new` statements to create two objects. The first is a `Random` object, like the one you used in the math quiz game. It is used in this code to randomly choose cells in the TableLayoutPanel. The second object, which may be new to you, is a `List<T>` object which is used to store the randomly-chosen symbols.

To add a random object and a list of icons

1. In **Solution Explorer**, choose *Form1.cs* if you're using C#, or *Form1.vb* if you're using Visual Basic, and then on the menu bar, choose **View > Code**. As an alternative, you can choose the **F7** key or double-click **Form1** in **Solution Explorer**.

This displays the code module behind Form1.

2. In the existing code, add the following code.

```
public partial class Form1 : Form
{
    // Use this Random object to choose random icons for the squares
    Random random = new Random();

    // Each of these letters is an interesting icon
    // in the Webdings font,
    // and each icon appears twice in this list
    List<string> icons = new List<string>()
    {
        "!", "!", "N", "N", ",", ",", "k", "k",
        "b", "b", "v", "v", "w", "w", "z", "z"
    };
}
```

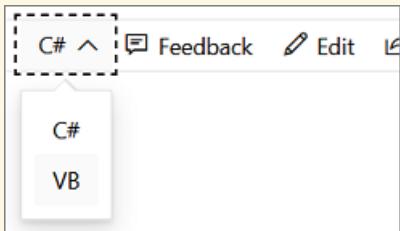
```
Public Class Form1

    ' Use this Random object to choose random icons for the squares
    Private random As New Random

    ' Each of these letters is an interesting icon
    ' in the Webdings font,
    ' and each icon appears twice in this list
    Private icons =
        New List(Of String) From {"!", "!", "N", "N", ",", ",", "k", "k",
                                "b", "b", "v", "v", "w", "w", "z", "z"}
```

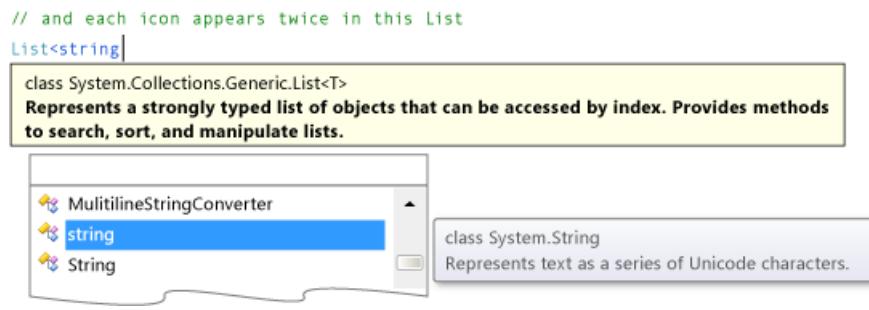
IMPORTANT

Use the programming language control at the top right of this page to view either the C# code snippet or the Visual Basic code snippet.



If you're using C#, be sure you put the code after the opening curly brace and just after the class declaration (`public partial class Form1 : Form`). If you're using Visual Basic, put the code right after the class declaration (`Public Class Form1`).

- When adding the List object, notice the **IntelliSense** window that opens. The following is a C# example, but similar text appears when you add a list in Visual Basic.



IntelliSense window

NOTE

The IntelliSense window appears only when you enter code manually. If you copy and paste the code, it doesn't appear.

If you look at the code (and remarks) in small sections, it's easier to understand. Your programs can use list objects to keep track of many different types of items. A list can hold numbers, true/false values, text, or other objects. You can even have a list object that holds other list objects. The items in a list are called elements, and each list only holds one type of element. So, a list of numbers can only hold numbers—you can't add text to that list. Similarly, you can't add numbers to a list of true/false values.

When you create a `List` object using a `new` statement, you need to specify the kind of data you want to store in it. That's why the tooltip at the top of the **IntelliSense** window shows the types of elements in the list. Also, that's what `List<string>` (in C#) and `List(Of String)` (in Visual Basic) means: It's a `List` object that holds elements of `string` data type. A string is what your program uses to store text, which is what the tooltip to the right of the **IntelliSense** window is telling you.

- Consider why in Visual Basic a temporary array must be created first, but in C#, the list can be created with one statement. This is because the C# language has *collection initializers*, which prepare the list to accept values. In Visual Basic, you can use a collection initializer. However, for compatibility with the previous version of Visual Basic, we recommend using the preceding code.

When you use a collection initializer with a `new` statement, after the new List object is created, the program fills it with the data you provided inside the curly braces. In this case, you get a list of strings named icons, and that list will be initialized so that it contains sixteen strings. Each of those strings is a single letter, and

they all correspond to the icons that will be in the labels. So, the game will have a pair of exclamation points, a pair of uppercase N letters, a pair of commas, and so on. (When these characters are set to the Webdings font, they will appear as symbols, such as a bus, a bike, a spider, and so forth.) Your list object will have sixteen strings in all, one for each cell in the TableLayoutPanel panel.

NOTE

In Visual Basic, you get the same result, but first the strings are put into a temporary array, which is then converted into a list object. An array is similar to a list, except, for example, arrays are created with a fixed size. Lists can shrink and grow as needed, which is important in this program.

To continue or review

- To go to the next tutorial step, see [Step 3: Assign a random icon to each label](#).
- To return to the previous tutorial step, see [Step 1: Create a project and add a table to your form](#).

Step 3: Assign a random icon to each label

10/18/2019 • 6 minutes to read • [Edit Online](#)

If the icons show up in the same cells every game, it's not very challenging. To avoid this, assign the icons randomly to the Label controls on your form by using an `AssignIconsToSquares()` method.

To assign a random icon to each label

1. Before adding the following code, consider how the method works. There's a new keyword: `foreach` in C# and `For Each` in Visual Basic. (One of the lines is commented out on purpose, which is explained at the end of this procedure.)

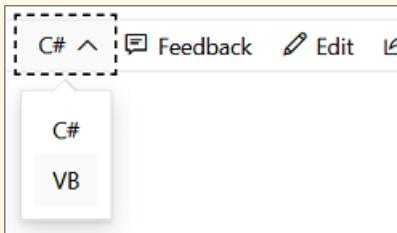
```
/// <summary>
/// Assign each icon from the list of icons to a random square
/// </summary>
private void AssignIconsToSquares()
{
    // The TableLayoutPanel has 16 labels,
    // and the icon list has 16 icons,
    // so an icon is pulled at random from the list
    // and added to each label
    foreach (Control control in tableLayoutPanel1.Controls)
    {
        Label iconLabel = control as Label;
        if (iconLabel != null)
        {
            int randomNumber = random.Next(Icons.Count);
            iconLabel.Text = Icons[randomNumber];
            // iconLabel.ForeColor = iconLabel.BackColor;
            Icons.RemoveAt(randomNumber);
        }
    }
}
```

```
''' <summary>
''' Assign each icon from the list of icons to a random square
''' </summary>
''' <remarks></remarks>
Private Sub AssignIconsToSquares()

    ' The TableLayoutPanel has 16 labels,
    ' and the icon list has 16 icons,
    ' so an icon is pulled at random from the list
    ' and added to each label
    For Each control In TableLayoutPanel1.Controls
        Dim iconLabel = TryCast(control, Label)
        If iconLabel IsNot Nothing Then
            Dim randomNumber = random.Next(Icons.Count)
            iconLabel.Text = Icons(randomNumber)
            ' iconLabel.ForeColor = iconLabel.BackColor
            Icons.RemoveAt(randomNumber)
        End If
    Next
End Sub
```

IMPORTANT

Use the programming language control at the top right of this page to view either the C# code snippet or the Visual Basic code snippet.



2. Add the `AssignIconsToSquares()` method as shown in the previous step. You can put it just below the code you added in [Step 2: Add a Random object and a list of icons](#).

As mentioned earlier, there's something new in your `AssignIconsToSquares()` method: a `foreach` loop in C# and `For Each` in Visual Basic. You can use a `For Each` loop any time you want to do the same action multiple times. In this case, you want to execute the same statements for every label on your `TableLayoutPanel`, as explained by the following code. The first line creates a variable named `control` that stores each control one at a time while that control has the statements in the loop executed on it.

```
foreach (Control control in tableLayoutPanel1.Controls)
{
    // The statements you want to execute
    // for each label go here
    // The statements use iconLabel to access
    // each label's properties and methods
}
```

```
For Each control In TableLayoutPanel1.Controls
    ' The statements you want to execute
    ' for each label go here
    ' The statements use iconLabel to access
    ' each label's properties and methods
Next
```

NOTE

The names "iconLabel" and "control" are used because they are descriptive. You can replace these names with any names, and the code will work exactly the same as long as you change the name in each statement inside the loop.

The `AssignIconsToSquares()` method iterates through each label control in the `TableLayoutPanel` and executes the same statements for each of them. Those statements pull a random icon from the list that you added in [Step 2: Add a Random object and a list of icons](#). (That's why you included two of each icon in the list, so there would be a pair of icons assigned to random Label controls.)

Look more closely at the code that runs inside the `foreach` or `For Each` loop. This code is reproduced here.

```
Label iconLabel = control as Label;
if (iconLabel != null)
{
    int randomNumber = random.Next(Icons.Count);
    iconLabel.Text = icons[randomNumber];
    // iconLabel.ForeColor = iconLabel.BackColor;
    icons.RemoveAt(randomNumber);
}
```

```
Dim iconLabel = TryCast(control, Label)
If iconLabel IsNot Nothing Then
    Dim randomNumber = random.Next(Icons.Count)
    iconLabel.Text = icons(randomNumber)
    ' iconLabel.ForeColor = iconLabel.BackColor
    icons.RemoveAt(randomNumber)
End If
```

The first line converts the **control** variable to a label named **iconLabel**. The line after that is an `if` statement that checks to make sure the conversion worked. If the conversion does work, the statements in the `if` statement run. (As you may recall from the previous tutorials, the `if` statement is used to evaluate whatever condition you specify.) The first line in the `if` statement creates a variable named **randomNumber** that contains a random number that corresponds to one of the items in the **icons** list. To do this, it uses the `Next()` method of the `Random` object that you created earlier. The `Next` method returns the random number. This line also uses the `Count` property of the **icons** list to determine the range from which to choose the random number. The next line assigns one of the icon list items to the `Text` property of the label. The commented-out line is explained later in this topic. Finally, the last line in the `if` statement removes from the list the icon that has been added to the form.

Remember, if you're not sure about what some part of the code does, you can position the mouse pointer over a code element and review the resulting tooltip. You can also step through each line of code while the program is running by using the Visual Studio debugger. See [How do I: Step with The debugger in Visual Studio?](#) or [Navigate through code with the debugger](#) for more information.

3. To fill up the game board with icons, you need to call the `AssignIconsToSquares()` method as soon as the program starts. If you're using C#, add a statement just below the call to the `InitializeComponent()` method in the **Form1** constructor, so your form calls your new method to set itself up before it's shown. Constructors are called when you create a new object, such as a class or struct. See [Constructors \(C# programming guide\)](#) or [Use constructors and destructors](#) in Visual Basic for more information.

```
public Form1()
{
    InitializeComponent();

    AssignIconsToSquares();
}
```

For Visual Basic, add the `AssignIconsToSquares()` method call to the `Form1_Load` method so that the code looks like the following.

```
Private Sub Form1_Load(sender As Object, e As EventArgs) Handles MyBase.Load
    AssignIconsToSquares()
End Sub
```

4. Save your program and run it. It should show a form with random icons assigned to each label.

5. Close your program, and then run it again. Notice that different icons are assigned to each label, as shown in the following picture.



Matching game with random icons

The icons are visible now because you haven't hidden them. To hide them from the player, you can set each label's **ForeColor** property to the same color as its **BackColor** property.

TIP

Another way to hide controls like labels is to set their **Visible** property to **False**.

6. To hide the icons, stop the program and remove the comment marks for the commented line of code inside the `For Each` loop.

```
iconLabel.ForeColor = iconLabel.BackColor;
```

```
iconLabel.ForeColor = iconLabel.BackColor
```

7. On the menu bar, choose the **Save All** button to save your program, and then run it. The icons seem to have disappeared—only a blue background appears. However, the icons are randomly assigned and are still there. Because the icons are the same color as the background, it hides them from the player. After all, it wouldn't be a very challenging game if the player could see all of the icons right away!

To continue or review

- To go to the next tutorial step, see [Step 4: Add a click event handler to each label](#).
- To return to the previous tutorial step, see [Step 2: Add a Random object and a list of icons](#).

Step 4: Add a click event handler to each label

10/18/2019 • 3 minutes to read • [Edit Online](#)

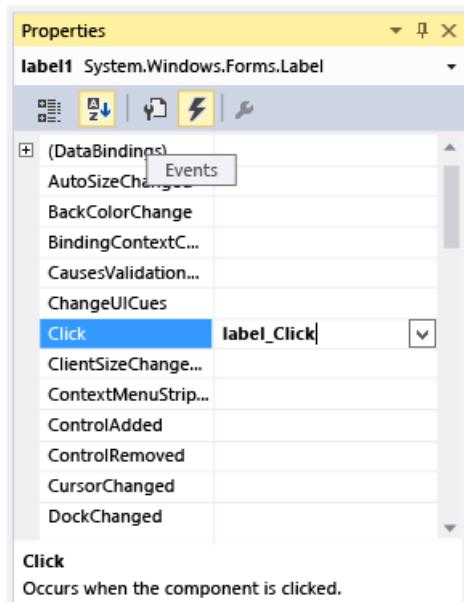
The matching game works as follows:

1. When a player chooses one of the squares with a hidden icon, the program shows the icon to the player by changing the icon color to black.
2. Then the player chooses another hidden icon.
3. If the icons match, they stay visible. If not, both icons are hidden again.

To get your program to work that way, you add a **Click** event handler that changes the color of the label that is chosen.

To add a click event handler to each label

1. Open the form in the **Windows Forms Designer**. In **Solution Explorer**, choose *Form1.cs* or *Form1.vb*. On the menu bar, choose **View > Designer**.
2. Choose the first label control to select it. Then, hold down the **Ctrl** key while you choose each of the other labels to select them. Be sure that every label is selected.
3. Choose the **Events** button on the tool bar in the **Properties** window to view the **Events** page in the **Properties** window. Scroll down to the **Click** event, and enter **label_Click** in the box, as shown in the following screenshot.



4. Choose the **Enter** key. The IDE adds a `Click` event handler called `label_Click()` to the code, and hooks it to each of the labels on the form.
5. Fill in the rest of the code, as follows:

```

/// <summary>
/// Every label's Click event is handled by this event handler
/// </summary>
/// <param name="sender">The label that was clicked</param>
/// <param name="e"></param>
private void label_Click(object sender, EventArgs e)
{
    Label clickedLabel = sender as Label;

    if (clickedLabel != null)
    {
        // If the clicked label is black, the player clicked
        // an icon that's already been revealed --
        // ignore the click
        if (clickedLabel.ForeColor == Color.Black)
            return;

        clickedLabel.ForeColor = Color.Black;
    }
}

```

```

''' <summary>
''' Every label's Click event is handled by this event handler
''' </summary>
''' <param name="sender">The label that was clicked</param>
''' <param name="e"></param>
''' <remarks></remarks>
Private Sub label_Click(ByVal sender As System.Object,
    ByVal e As System.EventArgs) Handles Label9.Click,
    Label8.Click, Label7.Click, Label6.Click, Label5.Click, Label4.Click,
    Label3.Click, Label2.Click, Label16.Click, Label15.Click, Label14.Click,
    Label13.Click, Label12.Click, Label11.Click, Label10.Click, Label1.Click

    Dim clickedLabel = TryCast(sender, Label)

    If clickedLabel IsNot Nothing Then

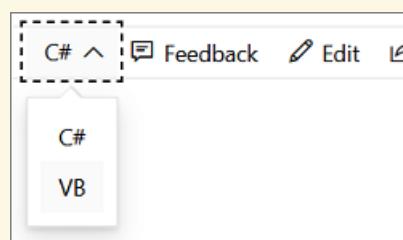
        ' If the clicked label is black, the player clicked
        ' an icon that's already been revealed --
        ' ignore the click
        If clickedLabel.ForeColor = Color.Black Then Exit Sub

        clickedLabel.ForeColor = Color.Black
    End If
End Sub

```

IMPORTANT

Use the programming language control at the top right of this page to view either the C# code snippet or the Visual Basic code snippet.



NOTE

If you copy and paste the `label_Click()` code block rather than entering the code manually, be sure to replace the existing `label_Click()` code. Otherwise, you'll end up with a duplicate code block.

NOTE

You may recognize `object sender` at the top of the event handler as the same one used in the [Tutorial 2: Create a timed math quiz](#) tutorial. Because you hooked up different label control click events to a single event handler method, the same method is called no matter which label the user chooses. The event handler method needs to know which label was chosen, so it uses the name `sender` to identify the label control. The first line of the method tells the program that it's not just a generic object, but specifically a label control, and that it uses the name `clickedLabel` to access the label's properties and methods.

This method first checks whether `clickedLabel` was successfully converted (cast) from an object to a label control. If unsuccessful, it has a value of `null` (C#) or `Nothing` (Visual Basic), and you don't want to execute the remainder of the code in the method. Next, the method checks the chosen label's text color by using the label's **ForeColor** property. If the label's text color is black, then that means the icon's already been chosen and the method is done. (That's what the `return` statement does: It tells the program to stop executing the method.) Otherwise, the icon hasn't been chosen, so the program changes the label's text color to black.

6. On the menu bar, choose **File** > **Save All** to save your progress, and then, on the menu bar, choose **Debug** > **Start Debugging** to run your program. You should see an empty form with a blue background. Choose any of the cells in the form, and one of the icons should become visible. Continue choosing different places in the form. As you choose the icons, they should appear.

To continue or review

- To go to the next tutorial step, see [Step 5: Add label references](#).
- To return to the previous tutorial step, see [Step 3: Assign a random icon to each label](#).

Step 5: Add label references

10/18/2019 • 4 minutes to read • [Edit Online](#)

The program needs to track which Label controls the player chooses. Right now, the program shows all labels chosen by the player. But we're going to change that. After the first label is chosen, the program should show the label's icon. After the second label is chosen, the program should display both icons for a brief time, and then hide both icons again. Your program will now keep track of which Label control is chosen first and which is chosen second by using *reference variables*.

To add label references

1. Add label references to your form by using the following code.

```
Public Class Form1

    ' firstClicked points to the first Label control
    ' that the player clicks, but it will be Nothing
    ' if the player hasn't clicked a label yet
    Private firstClicked As Label = Nothing

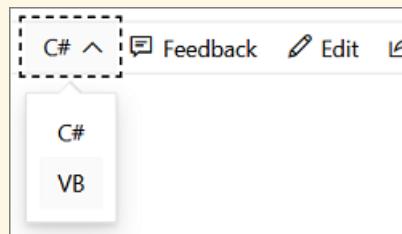
    ' secondClicked points to the second Label control
    ' that the player clicks
    Private secondClicked As Label = Nothing
```

```
public partial class Form1 : Form
{
    // firstClicked points to the first Label control
    // that the player clicks, but it will be null
    // if the player hasn't clicked a label yet
    Label firstClicked = null;

    // secondClicked points to the second Label control
    // that the player clicks
    Label secondClicked = null;
```

IMPORTANT

Use the programming language control at the top right of this page to view either the C# code snippet or the Visual Basic code snippet.



These reference variables look similar to the statements you used earlier to add objects (like `Timer` objects, `List<T>` objects, and `Random` objects) to your form. However, these statements don't cause two extra Label controls to appear on the form because there's no `new` keyword used in either of the two statements. Without the `new` keyword, no object is created. That's why `firstClicked` and `secondClicked` are called

reference variables: They just keep track (or, refer to) Label objects.

When a variable isn't keeping track of an object, it's set to a special reserved value: `null` in C# and `Nothing` in Visual Basic. So, when the program starts, both `firstClicked` and `secondClicked` are set to `null` or `Nothing`, which means that the variables aren't keeping track of anything.

2. Modify your `Click` event handler to use the new `firstClicked` reference variable. Remove the last statement in the `label_Click()` event handler method (`clickedLabel.ForeColor = Color.Black;`) and replace it with the `if` statement that follows. (Be sure you include the comment, and the whole `if` statement.)

```
''' <summary>
''' Every label's Click event is handled by this event handler
''' </summary>
''' <param name="sender">The label that was clicked</param>
''' <param name="e"></param>
''' <remarks></remarks>
Private Sub label_Click(ByVal sender As System.Object,
    ByVal e As System.EventArgs) Handles Label9.Click,
    Label8.Click, Label7.Click, Label6.Click, Label5.Click, Label4.Click,
    Label3.Click, Label2.Click, Label16.Click, Label15.Click, Label14.Click,
    Label13.Click, Label12.Click, Label11.Click, Label10.Click, Label1.Click

    Dim clickedLabel = TryCast(sender, Label)

    If clickedLabel IsNot Nothing Then

        ' If the clicked label is black, the player clicked
        ' an icon that's already been revealed --
        ' ignore the click
        If clickedLabel.ForeColor = Color.Black Then Exit Sub

        ' If firstClicked is Nothing, this is the first icon
        ' in the pair that the player clicked,
        ' so set firstClicked to the label that the player
        ' clicked, change its color to black, and return
        If firstClicked Is Nothing Then
            firstClicked = clickedLabel
            firstClicked.ForeColor = Color.Black
            Exit Sub
        End If
    End If

End Sub
```

```

/// <summary>
/// Every label's Click event is handled by this event handler
/// </summary>
/// <param name="sender">The label that was clicked</param>
/// <param name="e"></param>
private void label_Click(object sender, EventArgs e)
{
    Label clickedLabel = sender as Label;

    if (clickedLabel != null)
    {
        // If the clicked label is black, the player clicked
        // an icon that's already been revealed --
        // ignore the click
        if (clickedLabel.ForeColor == Color.Black)
            return;

        // If firstClicked is null, this is the first icon
        // in the pair that the player clicked,
        // so set firstClicked to the label that the player
        // clicked, change its color to black, and return
        if (firstClicked == null)
        {
            firstClicked = clickedLabel;
            firstClicked.ForeColor = Color.Black;

            return;
        }
    }
}

```

3. Save and run your program. Choose one of the label controls, and its icon appears.
4. Choose the next label control, and notice that nothing happens. The program is already keeping track of the first label that the player chose, so `firstClicked` isn't equal to `null` in C# or `Nothing` in Visual Basic. When your `if` statement checks `firstClicked` to determine if it's equal to `null` or `Nothing`, it finds that it isn't, and it doesn't execute the statements in the `if` statement. So, only the first icon that's chosen turns black, and the other icons are invisible, as shown in the following image.



Matching game showing one icon

You'll fix this situation in the next step of the tutorial by adding a **Timer** control.

To continue or review

- To go to the next tutorial step, see [Step 6: Add a timer](#).
- To return to the previous tutorial step, see [Step 4: Add a Click event handler to each label](#).

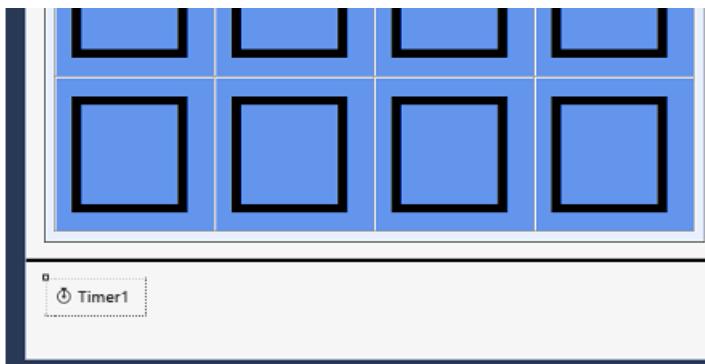
Step 6: Add a timer

10/18/2019 • 7 minutes to read • [Edit Online](#)

Next, you add a **Timer** control to the matching game. A timer waits a specified number of milliseconds, and then fires an event, referred to as a *tick*. This is useful for starting an action, or repeating an action on a regular basis. In this case, you'll use a timer to enable players to choose two icons, and if the icons don't match, hide the two icons again after a short period of time.

To add a timer

1. From the toolbox in **Windows Forms Designer**, choose **Timer** (in the **Components** category) and then choose the **Enter** key, or double-click the timer to add a timer control to the form. The timer's icon, called **Timer1**, should appear in a space below the form, as shown in the following image.



Timer

NOTE

If the toolbox is empty, be sure to select the form designer, and not the code behind the form, before opening the toolbox.

2. Choose the **Timer1** icon to select the timer. In the **Properties** window, switch from viewing events to viewing properties. Then, set the timer's **Interval** property to **750**, but leave its **Enabled** property set to **False**. The **Interval** property tells the timer how long to wait between *ticks*, or when it triggers its **Tick** event. A value of 750 tells the timer to wait three quarters of a second (750 milliseconds) before it fires its Tick event. You'll call the **Start()** method to start the timer only after the player chooses the second label.
3. Choose the timer control icon in **Windows Forms Designer** and then choose the **Enter** key, or double-click the timer, to add an empty Tick event handler. Either replace the code with the following code, or manually enter the following code into the event handler.

```

/// <summary>
/// This timer is started when the player clicks
/// two icons that don't match,
/// so it counts three quarters of a second
/// and then turns itself off and hides both icons
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>
private void timer1_Tick(object sender, EventArgs e)
{
    // Stop the timer
    timer1.Stop();

    // Hide both icons
    firstClicked.ForeColor = firstClicked.BackColor;
    secondClicked.ForeColor = secondClicked.BackColor;

    // Reset firstClicked and secondClicked
    // so the next time a label is
    // clicked, the program knows it's the first click
    firstClicked = null;
    secondClicked = null;
}

```

```

''' <summary>
''' This timer is started when the player clicks
''' two icons that don't match,
''' so it counts three quarters of a second
''' and then turns itself off and hides both icons
''' </summary>
''' <remarks></remarks>
Private Sub Timer1_Tick() Handles Timer1.Tick

    ' Stop the timer
    Timer1.Stop()

    ' Hide both icons
    firstClicked.ForeColor = firstClicked.BackColor
    secondClicked.ForeColor = secondClicked.BackColor

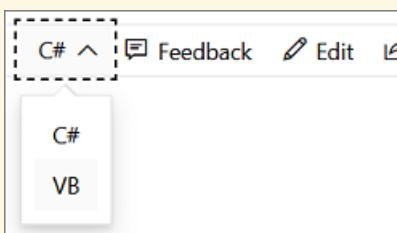
    ' Reset firstClicked and secondClicked
    ' so the next time a label is
    ' clicked, the program knows it's the first click
    firstClicked = Nothing
    secondClicked = Nothing

End Sub

```

IMPORTANT

Use the programming language control at the top right of this page to view either the C# code snippet or the Visual Basic code snippet.



The Tick event handler does three things: First, it makes sure the timer isn't running by calling the `Stop()` method. Then it uses two reference variables, `firstClicked` and `secondClicked`, to make the icons of the two labels that the player chose invisible again. Finally, it resets the `firstClicked` and `secondClicked` reference variables to `null` in C# and `Nothing` in Visual Basic. This step is important because it's how the program resets itself. Now it's not keeping track of any `Label` controls, and it's ready for the player to choose a label again.

NOTE

A Timer object has a `Start()` method that starts the timer, and a `Stop()` method that stops it. When you set the timer's **Enabled** property to **True** in the **Properties** window, it starts ticking as soon as the program begins. But when you leave it set to **False**, it doesn't start ticking until its `Start()` method is called. Normally, a timer fires its Tick event over and over again, using the **Interval** property to determine how many milliseconds to wait between ticks. You may have noticed how the timer's `Stop()` method is called inside the Tick event. That puts the timer into *one shot mode*, meaning that when the `Start()` method is called, it waits for the specified interval, triggers a single Tick event, and then stops.

4. To see the new timer in action, go to the code editor and add the following code to the top and bottom of the `label_Click()` event handler method. (You're adding an `if` statement to the top, and three statements to the bottom; the rest of the method stays the same.)

```
/// <summary>
/// Every label's Click event is handled by this event handler
/// </summary>
/// <param name="sender">The label that was clicked</param>
/// <param name="e"></param>
private void label_Click(object sender, EventArgs e)
{
    // The timer is only on after two non-matching
    // icons have been shown to the player,
    // so ignore any clicks if the timer is running
    if (timer1.Enabled == true)
        return;

    Label clickedLabel = sender as Label;

    if (clickedLabel != null)
    {
        // If the clicked label is black, the player clicked
        // an icon that's already been revealed --
        // ignore the click
        if (clickedLabel.ForeColor == Color.Black)
            return;

        // If firstClicked is null, this is the first icon
        // in the pair that the player clicked,
        // so set firstClicked to the label that the player
        // clicked, change its color to black, and return
        if (firstClicked == null)
        {
            firstClicked = clickedLabel;
            firstClicked.ForeColor = Color.Black;
            return;
        }

        // If the player gets this far, the timer isn't
        // running and firstClicked isn't null,
        // so this must be the second icon the player clicked
        // Set its color to black
        secondClicked = clickedLabel;
        secondClicked.ForeColor = Color.Black;

        // If the player gets this far, the player
        // clicked two different icons, so start the
        // timer (which will wait three quarters of
        // a second, and then hide the icons)
        timer1.Start();
    }
}
```

```

''' <summary>
''' Every label's Click event is handled by this event handler
''' </summary>
''' <param name="sender">The label that was clicked</param>
''' <param name="e"></param>
''' <remarks></remarks>
Private Sub label_Click(ByVal sender As System.Object,
                       ByVal e As System.EventArgs) Handles Label9.Click,
Label8.Click, Label7.Click, Label6.Click, Label5.Click, Label4.Click,
Label3.Click, Label2.Click, Label16.Click, Label15.Click, Label14.Click,
Label13.Click, Label12.Click, Label11.Click, Label10.Click, Label1.Click

    ' The timer is only on after two non-matching
    ' icons have been shown to the player,
    ' so ignore any clicks if the timer is running
    If Timer1.Enabled Then Exit Sub

    Dim clickedLabel = TryCast(sender, Label)

    If clickedLabel Is Not Nothing Then
        ' If the clicked label is black, the player clicked
        ' an icon that's already been revealed --
        ' ignore the click
        If clickedLabel.ForeColor = Color.Black Then Exit Sub

        ' If firstClicked is Nothing, this is the first icon
        ' in the pair that the player clicked,
        ' so set firstClicked to the label that the player
        ' clicked, change its color to black, and return
        If firstClicked Is Nothing Then
            firstClicked = clickedLabel
            firstClicked.ForeColor = Color.Black
            Exit Sub
        End If

        ' If the player gets this far, the timer isn't
        ' running and firstClicked isn't Nothing,
        ' so this must be the second icon the player clicked
        ' Set its color to black
        secondClicked = clickedLabel
        secondClicked.ForeColor = Color.Black

        ' If the player gets this far, the player
        ' clicked two different icons, so start the
        ' timer (which will wait three quarters of
        ' a second, and then hide the icons)
        Timer1.Start()
    End If

End Sub

```

The code at the top of the method checks whether the timer was started by checking the value of the **Enabled** property. That way, if the player chooses the first and second Label controls and the timer starts, choosing a third label won't do anything.

The code at the bottom of the method sets the `secondClicked` reference variable to track the second Label control that the player chose, and then it sets that label's icon color to black to make it visible. Then, it starts the timer in one shot mode, so that it waits 750 milliseconds and then fires a single Tick event. The timer's Tick event handler hides the two icons and resets the `firstClicked` and `secondClicked` reference variables so the form is ready for the player to choose another pair of icons.

5. Save and run your program. Choose an icon, and it becomes visible.
6. Choose another icon. It appears briefly, and then both icons disappear. Repeat this numerous times. The

form now keeps track of the first and second icons that you choose, and uses the timer to pause before making the icons disappear.

To continue or review

- To go to the next tutorial step, see [Step 7: Keep pairs visible](#).
- To return to the previous tutorial step, see [Step 5: Add label references](#).

Step 7: Keep pairs visible

10/18/2019 • 3 minutes to read • [Edit Online](#)

The game works well, as long as the player only chooses pairs of icons that don't match. But consider what should happen when the player chooses a matching pair. Instead of making the icons disappear by turning on the timer (using the `Start()` method), the game should reset itself so that it's no longer keeping track of any labels using the `firstClicked` and `secondClicked` reference variables, without resetting the colors for the two labels that were chosen.

To keep pairs visible

1. Add the following `if` statement to the `label_Click()` event handler method, near the end of the code just above the statement where you start the timer. Take a close look at the code while adding it to the program. Consider how the code works.

```
// If the player gets this far, the timer isn't
// running and firstClicked isn't null,
// so this must be the second icon the player clicked
// Set its color to black
secondClicked = clickedLabel;
secondClicked.ForeColor = Color.Black;

// If the player clicked two matching icons, keep them
// black and reset firstClicked and secondClicked
// so the player can click another icon
if (firstClicked.Text == secondClicked.Text)
{
    firstClicked = null;
    secondClicked = null;
    return;
}

// If the player gets this far, the player
// clicked two different icons, so start the
// timer (which will wait three quarters of
// a second, and then hide the icons)
timer1.Start();
}
```

```

' If the player gets this far, the timer isn't
' running and firstClicked isn't Nothing,
' so this must be the second icon the player clicked
' Set its color to black
secondClicked = clickedLabel
secondClicked.ForeColor = Color.Black

' If the player clicked two matching icons, keep them
' black and reset firstClicked and secondClicked
' so the player can click another icon
If firstClicked.Text = secondClicked.Text Then
    firstClicked = Nothing
    secondClicked = Nothing
    Exit Sub
End If

' If the player gets this far, the player
' clicked two different icons, so start the
' timer (which will wait three quarters of
' a second, and then hide the icons)
Timer1.Start()
End If
End Sub

```

> [!IMPORTANT]
 > Use the programming language control at the top right of this page to view either the C# code snippet or the Visual Basic code snippet.

![Programming language control for Docs.Microsoft.com]
 (../ide/media/docs-programming-language-control.png)

The first line of the `if` statement you just added checks whether the icon in the first label that the player chooses is the same as the icon in the second label. If the icons are identical, the program executes the three statements between the curly braces in C# or the three statements within the `if` statement in Visual Basic. The first two statements reset the `firstClicked` and `secondClicked` reference variables so that they no longer keep track of any of the labels. (You may recognize those two statements from the timer's `Tick` event handler.) The third statement is a `return` statement, which tells the program to skip the rest of the statements in the method without executing them.

If programming in C#, you may have noticed that some of the code uses a single equal sign (`=`), while other statements use two equal signs (`==`). Consider why `=` is used in some places but `==` is used in other places.

This is a good example that shows the difference. Take a careful look at the code between the parentheses in the `if` statement.

```
firstClicked.Text = secondClicked.Text
```

```
firstClicked.Text == secondClicked.Text
```

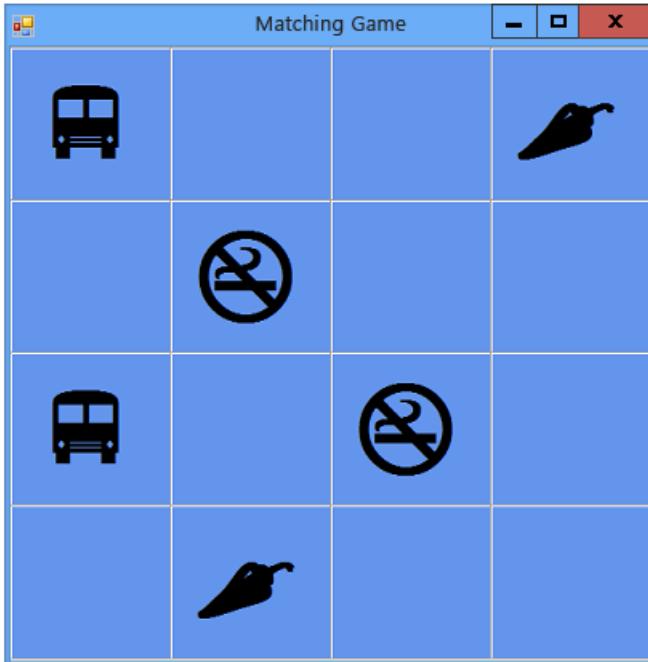
Then look closely at the first statement in the block of code after the `if` statement.

```
firstClicked = Nothing
```

```
firstClicked = null;
```

The first of those two statements checks whether two icons are the same. Because two values are being compared, the C# program uses the `==` equality operator. The second statement actually changes the value (called *assignment*), setting the `firstClicked` reference variable equal to `null` to reset it. That's why it uses the `=` assignment operator instead. C# uses `=` to set values, and `==` to compare them. Visual Basic uses `=` for both variable assignment and comparison.

2. Save and run the program, and then start choosing icons on the form. If you choose a pair that doesn't match, the timer's Tick event triggers, and both icons disappear. If you choose a matching pair, the new `if` statement executes, and the return statement causes the method to skip the code that starts the timer, so the icons stay visible, as shown in the following image.



Matching game with visible icon pairs

To continue or review

- To go to the next tutorial step, see [Step 8: Add a method to verify whether the player won](#).
- To return to the previous tutorial step, see [Step 6: Add a timer](#).

Step 8: Add a method to verify whether the player won

10/18/2019 • 3 minutes to read • [Edit Online](#)

You've created a fun game, but it needs an additional item to finish it. The game should end when the player wins, so you need to add a `CheckForWinner()` method to verify whether the player won.

To add a method to verify whether the player won

1. Add a `CheckForWinner()` method to the bottom of your code, below the `timer1_Tick()` event handler, as shown in the following code.

```
/// <summary>
/// Check every icon to see if it is matched, by
/// comparing its foreground color to its background color.
/// If all of the icons are matched, the player wins
/// </summary>
private void CheckForWinner()
{
    // Go through all of the labels in the TableLayoutPanel,
    // checking each one to see if its icon is matched
    foreach (Control control in tableLayoutPanel1.Controls)
    {
        Label iconLabel = control as Label;

        if (iconLabel != null)
        {
            if (iconLabel.ForeColor == iconLabel.BackColor)
                return;
        }
    }

    // If the loop didn't return, it didn't find
    // any unmatched icons
    // That means the user won. Show a message and close the form
    MessageBox.Show("You matched all the icons!", "Congratulations");
    Close();
}
```

```

''' <summary>
''' Check every icon to see if it is matched, by
''' comparing its foreground color to its background color.
''' If all of the icons are matched, the player wins
''' </summary>
Private Sub CheckForWinner()

    ' Go through all of the labels in the TableLayoutPanel,
    ' checking each one to see if its icon is matched
    For Each control In TableLayoutPanel1.Controls
        Dim iconLabel = TryCast(control, Label)
        If iconLabel IsNot Nothing AndAlso
            iconLabel.ForeColor = iconLabel.BackColor Then Exit Sub
    Next

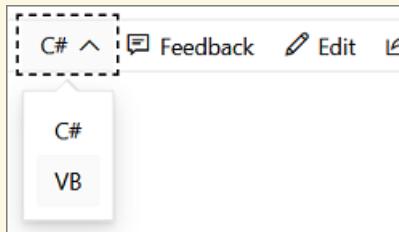
    ' If the loop didn't return, it didn't find
    ' any unmatched icons
    ' That means the user won. Show a message and close the form
    MessageBox.Show("You matched all the icons!", "Congratulations")
    Close()

End Sub

```

IMPORTANT

Use the programming language control at the top right of this page to view either the C# code snippet or the Visual Basic code snippet.



The method uses another `foreach` loop in C# or `For Each` loop in Visual Basic to go through each label in the `TableLayoutPanel`. It uses the equality operator (`==` in C# and `=` in Visual Basic) to check each label's icon color to verify whether it matches the background. If the colors match, the icon remains invisible, and the player hasn't matched all of the icons remaining. In that case, the program uses a `return` statement to skip the rest of the method. If the loop gets through all of the labels without executing the `return` statement, that means that all of the icons on the form were matched. The program shows a `MessageBox` to congratulate the player on winning, and then calls the form's `close()` method to end the game.

2. Next, have the label's `Click` event handler call the new `CheckForWinner()` method. Be sure that your program checks for a winner immediately after it shows the second icon that the player chooses. Look for the line where you set the second chosen icon's color, and then call the `CheckForWinner()` method right after that, as shown in the following code.

```

// If the player gets this far, the timer isn't
// running and firstClicked isn't null,
// so this must be the second icon the player clicked
// Set its color to black
secondClicked = clickedLabel;
secondClicked.ForeColor = Color.Black;

// Check to see if the player won
CheckForWinner();

// If the player clicked two matching icons, keep them
// black and reset firstClicked and secondClicked
// so the player can click another icon
if (firstClicked.Text == secondClicked.Text)
{
    firstClicked = null;
    secondClicked = null;
    return;
}

```

```

' If the player gets this far, the timer isn't
' running and firstClicked isn't Nothing,
' so this must be the second icon the player clicked
' Set its color to black
secondClicked = clickedLabel
secondClicked.ForeColor = Color.Black

' Check to see if the player won
CheckForWinner()

' If the player clicked two matching icons, keep them
' black and reset firstClicked and secondClicked
' so the player can click another icon
If firstClicked.Text = secondClicked.Text Then
    firstClicked = Nothing
    secondClicked = Nothing
    Exit Sub
End If

```

- Save and run the program. Play the game and match all of the icons. When you win, the program displays a congratulatory **MessageBox** (as shown in the following screenshot), and then closes the box.



Matching game with MessageBox

To continue or review

- To go to the next tutorial step, see [**Step 9: Try other features.**](#)
- To return to the previous tutorial step, see [**Step 7: Keep pairs visible.**](#)

Step 9: Try other features

10/25/2019 • 2 minutes to read • [Edit Online](#)

To learn more, try changing icons and colors, adding a game timer, and adding sounds. To make the game more challenging, try making the board bigger and adjusting the timer.

To download a completed version of the sample, see [Complete matching game tutorial sample](#).

To try other features

- Replace the icons and colors with ones you choose.

TIP

Try looking at the label's `Forecolor` property.

- Add a game timer that tracks how long it takes for the player to win.

TIP

To do this, you can add a label to display the elapsed time on the form above the `TableLayoutPanel`, and add another timer to the form to track the time. Use code to start the timer when the player starts the game, and stop the timer after they match the last two icons.

- Add a sound when the player finds a match, another sound when the player uncovers two icons that don't match, and a third sound when the program hides the icons again.

TIP

To play sounds, you can use the `System.Media` namespace. See [Play sounds in Windows Forms app \(C#\)](#) or [How to play audio in Visual Basic](#) for more information.

- Make the game more difficult by making the board bigger.

TIP

You'll need to do more than just add rows and columns to the `TableLayoutPanel` - you'll also need to consider the number of icons you create.

- Make the game more challenging by hiding the first icon if the player is too slow to respond and doesn't choose the second icon before a certain amount of time.

To continue or review

- There are great, free video learning resources available to you. To learn more about programming in Visual Basic, see [Visual Basic fundamentals: Development for absolute beginners](#). To learn more about programming in C#, see [C# fundamentals: Development for absolute beginners](#).
- To return to the previous tutorial step, see [Step 8: Add a method to verify whether the player won](#).

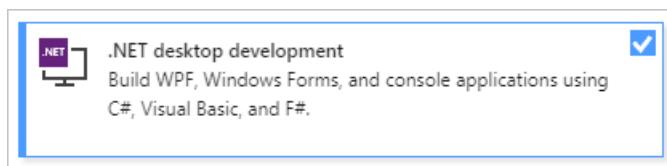
Develop with Visual F# in Visual Studio

10/18/2019 • 7 minutes to read • [Edit Online](#)

This article includes information about Visual Studio features for F# development.

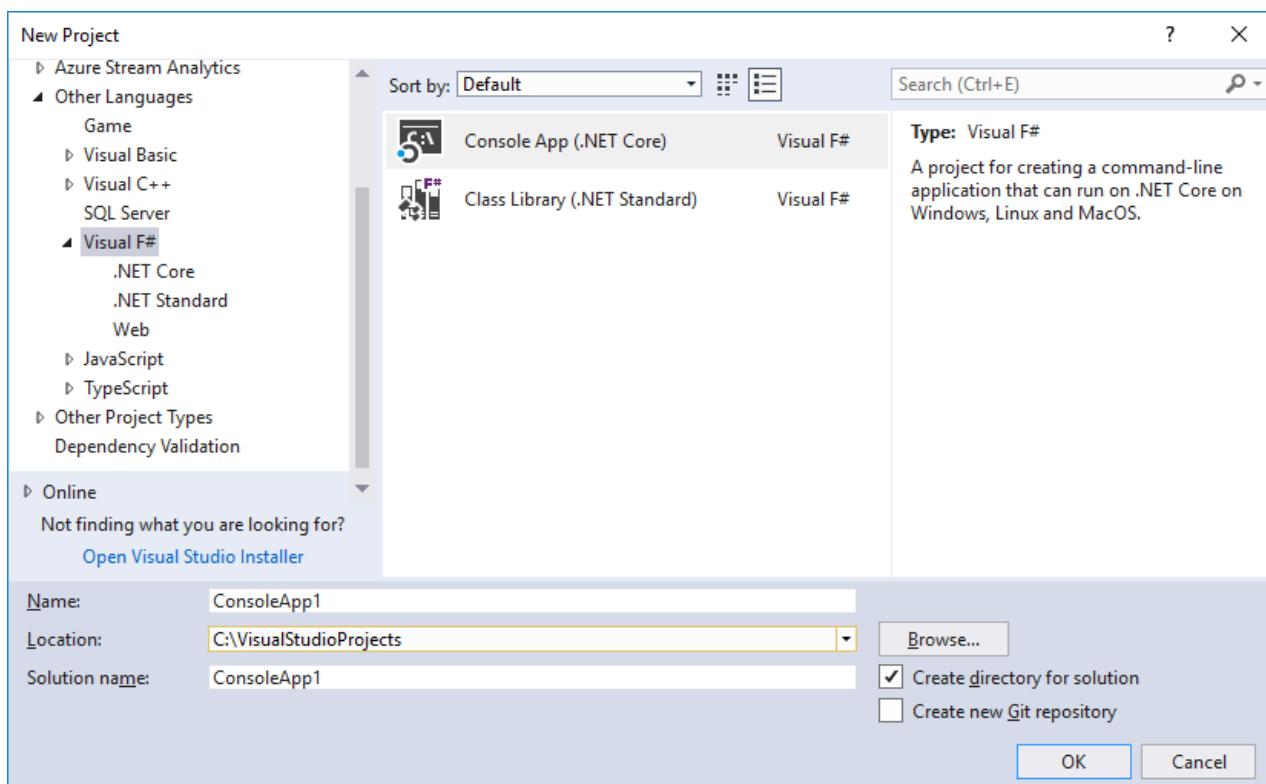
Install F# support

To develop with F# in Visual Studio, first install the **.NET desktop development** workload if you haven't already. You install Visual Studio workloads through Visual Studio Installer, which you can open by selecting **Tools > Get Tools and Features**.

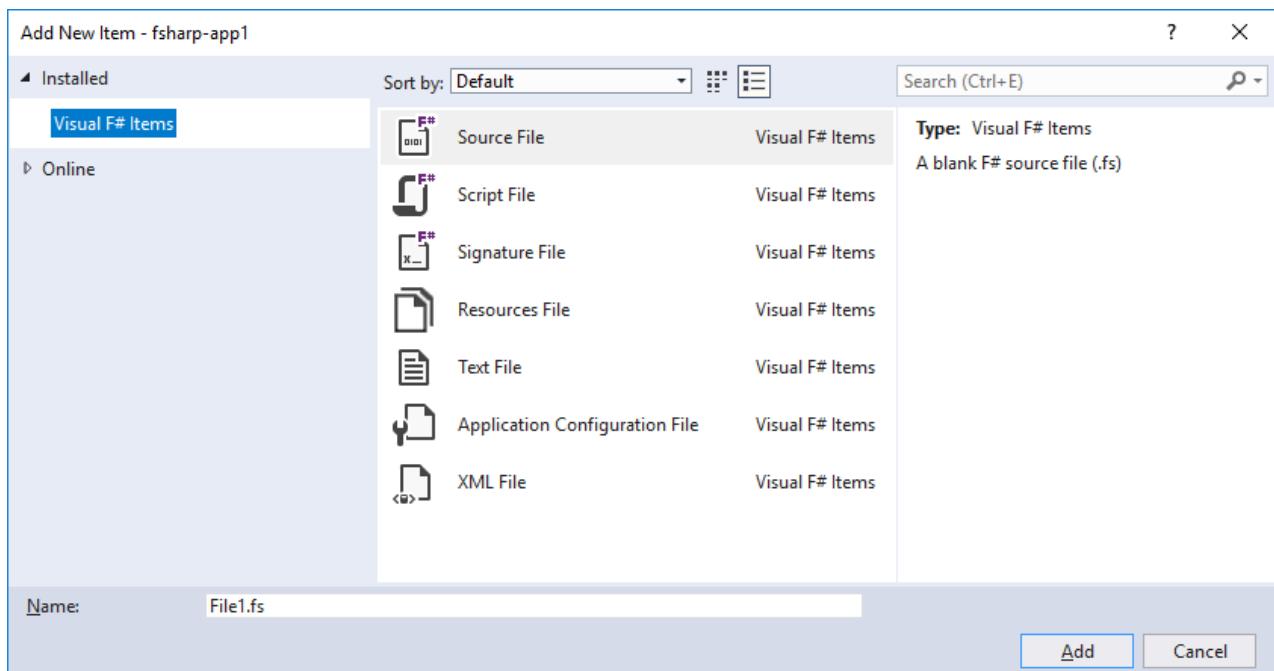


F# project features

Various project and item templates are available for F# in Visual Studio. The following image shows some of the F# project templates for .NET Core and .NET Standard:



The following image shows some of the F# item templates:



For more information about the item templates for data access, see [F# type providers](#).

The following table summarizes features in project properties for F#:

PROJECT SETTING	SUPPORTED IN F#?	NOTES
Resource files	Yes	
Build, debug, and reference settings	Yes	
Multitargeting	Yes	
Icon and manifest	No	Available through compiler command-line options.
ASP.NET Client Services	No	
ClickOnce	No	Use a client project in another .NET language, if applicable.
Strong naming	No	Available through compiler command-line options.
Assembly publishing and versioning	No	
Code analysis	No	Code analysis tools can be run manually or as part of a post-build command.
Security (change trust levels)	No	

Project Designer

Project Designer consists of several project property pages grouped by related functionality. The pages available for F# projects are mostly a subset of those available for other languages, and are described in the following table. Links are provided to the corresponding C# **Project Designer** page.

PROJECT DESIGNER PAGE	RELATED LINKS	DESCRIPTION
Application	Application Page, Project Designer	Enables you to specify application-level settings and properties, such as whether you are creating a library or an executable file, what version of .NET the application targets, and information about where the resource files that the application uses are stored.
Build	Build Page, Project Designer	Enables you to control how the code is compiled.
Build Events	Build Events Page, Project Designer	Enables you to specify commands to run before or after a compilation.
Debug	Debug Page, Project Designer	Enables you to control how the application runs during debugging. This includes what commands to use and what your application's starting directory is, and any special debugging modes you want to enable, such as native code and SQL.
Package (.NET SDK only)	N/A	Enables you to define NuGet Package metadata when publishing as a NuGet package.
Reference Paths	Manage references in a project	Enables you to specify where to search for assemblies that the code depends on.
Resources (.NET SDK only)	N/A	Enables you to generate and manage a default resources file.

F#-specific settings

The following table summarizes settings that are specific to F#:

PROJECT DESIGNER PAGE	SETTING	DESCRIPTION
Build	Generate tail calls	If selected, enables the use of the tail Microsoft Intermediate Language (MSIL) instruction. This causes the stack frame to be reused for tail recursive functions. Equivalent to the <code>--tailcalls</code> compiler option.
Build	Other flags	Allows you to specify additional compiler command-line options.

Code and text editor features

The following features of the Visual Studio code and text editors are supported in F#:

FEATURE	DESCRIPTION	SUPPORTED IN F#?
Automatically comment	Enables you to comment or uncomment sections of code.	Yes
Automatically format	Reformats code with standard indentation and style.	No
Bookmarks	Enables you to save places in the editor.	Yes
Change indentation	Indents or unindents selected lines.	Yes
Smart indentation	Automatically indents and de-indents the cursor according to F# scoping rules.	Yes
Find and replace text	Enables you to search in a file, project, or solution, and potentially change text.	Yes
Go to definition for the .NET API	When the cursor is positioned on a .NET API, shows code generated from .NET metadata.	No
Go to definition for user-defined API	When the cursor is on a program entity that you defined, moves the cursor to the location in your code where the entity is defined.	Yes
Go To Line	Enables you to go to a specific line in a file, by line number.	Yes
Navigation bars at top of file	Enables you to jump to locations in code, by, For example, function name.	Yes
Block Structure Guidelines	Shows guidelines that indicate F# scopes, which can be hovered over for a preview.	Yes
Outlining	Enables you to collapse sections of your code to create a more compact view.	Yes
Tabify	Converts spaces to tabs.	Yes
Type colorization	Shows defined type names in a special color.	Yes
Quick Find. See Quick Find, Find and Replace Window.	Enables you to search in a file or project.	Yes
Ctrl+click to Go to Definition	Allows you to hold Ctrl and click on an F# symbol to invoke Go to Definition.	Yes
Go to Definition from QuickInfo	Clickable symbols inside tooltips that invoke Go to Definition.	Yes

FEATURE	DESCRIPTION	SUPPORTED IN F#?
Go to All	Enables global, fuzzy-matching navigation for all F# constructs via Ctrl+T .	Yes
Inline Rename	Renames all occurrences of a symbol inline.	Yes
Find all References	Finds all occurrences of a symbol in a codebase.	Yes
Simplify Name code fix	Removes unnecessary qualifiers for F# symbols.	Yes
Remove unused <code>open</code> statement code fix	Removes all unnecessary <code>open</code> statements in a document.	Yes
Unused value code fix	Suggests renaming an unused identifier to underscore.	Yes

For general information about editing code in Visual Studio, and features of the text editor, see [Write code in the editor](#).

IntelliSense features

The following table summarizes IntelliSense features supported and not supported in F#:

FEATURE	DESCRIPTION	SUPPORTED IN F#?
Automatically implement interfaces	Generates code stubs for interface methods.	Yes
Code snippets	Injects code from a library of common coding constructs into topics.	No
Complete Word	Saves typing by completing words and names as you type.	Yes
Automatic completion	When enabled, causes the word completion to select the first match as you type, instead of waiting for you to select one or press Ctrl+Space .	Yes
Offer completion for symbols in unopened namespaces	With automatic completion, a matching symbol that resides in an unopened namespace is suggested, offering to complete with the corresponding <code>open</code> statement when selected.	Yes
Generate code elements	Enables you to generate stub code for a variety of constructs.	No
List Members	When you type the member access operator (.), shows members for a type.	Yes

FEATURE	DESCRIPTION	SUPPORTED IN F#?
Organize Usings/Open	Organizes namespaces referenced by using statements in C# or open directives in F#.	No
Parameter Info	Shows helpful information about parameters as you type a function call.	Yes
Quick Info	Displays the complete declaration for any identifier in your code.	Yes
Automatic brace completion	Automatically completes F# brace-like syntax constructs in a transactional manner.	Yes

For general information about IntelliSense, see [Use IntelliSense](#).

Debugging features

The following table summarizes features that are available when you debug F# code:

FEATURE	DESCRIPTION	SUPPORTED IN F#?
Autos window	Shows automatic or temporary variables.	No
Breakpoints	Enables you to pause code execution at specific points during debugging.	Yes
Conditional breakpoints	Enables breakpoints that test a condition that determines whether execution should pause.	Yes
Edit and Continue	Enables code to be modified and compiled as you debug a running program without stopping and restarting the debugger.	No
Expression evaluator	Evaluates and executes code at run time.	No, but the C# expression evaluator can be used, although you must use C# syntax.
Historical debugging	Enables you to step into previously executed code.	Yes
Locals window	Shows locally defined values and variables.	Yes
Run To Cursor	Enables you to execute code until the line that contains the cursor is reached.	Yes
Step Into	Enables you to advance execution and move into any function call.	Yes

FEATURE	DESCRIPTION	SUPPORTED IN F#?
Step Over	Enables you to advance execution in the current stack frame and move past any function call.	Yes

For general information about the Visual Studio debugger, see [Debugging in Visual Studio](#).

Additional tools

The following table summarizes the support for F# in Visual Studio tools.

TOOL	DESCRIPTION	SUPPORTED IN F#?
Call Hierarchy	Displays the nested structure of function calls in your code.	No
Code Metrics	Gathers information about your code, such as line counts.	No
Class View	Provides a type-based view of the code in a project.	No
Error List window	Shows a list of errors in code.	Yes
F# Interactive	Enables you to type (or copy and paste) F# code and run it immediately, independently of the building of your project. The F# Interactive window is a Read, Evaluate, Print Loop (REPL).	Yes
Object Browser	Enables you to view the types in an assembly.	F# types as they appear in compiled assemblies do not appear exactly as you author them. You can browse through the compiled representation of F# types, but you cannot view the types as they appear from F#.
Output window	Displays build output.	Yes
Performance analysis	Provides tools for measuring the performance of your code.	Yes
Properties window	Displays and enables editing of properties of the object in the development environment that has focus.	Yes
Server Explorer	Provides ways to interact with a variety of server resources.	Yes
Solution Explorer	Enables you to view and manage projects and files.	Yes
Task List	Enables you to manage work items pertaining to your code.	No

TOOL	DESCRIPTION	SUPPORTED IN F#?
Test projects	Provides features that help you test your code.	No
Toolbox	Displays tabs that contain draggable objects such as controls and sections of text or code.	Yes

See also

- [F# guide \(.NET Framework\)](#)
- [Get started with F# in Visual Studio](#)

Target older versions of .NET (F#)

10/18/2019 • 2 minutes to read • [Edit Online](#)

The following error might appear if you try to target the .NET Framework 2.0, 3.0, or 3.5 in an F# project when Visual Studio is installed on Windows 8.1:

This project requires the 2.0 F# runtime, but that runtime is not installed.

This error is known to occur under the following combination of conditions:

- You installed Visual Studio on Windows 8.1.
- You didn't enable the .NET Framework 3.5 before you installed Visual Studio.
- Your project targets the .NET Framework 2.0, 3.0, or 3.5.

When you install Visual Studio, it detects the installed versions of the .NET Framework. Visual Studio installs the F# 2.0 runtime only if the .NET Framework 3.5 is installed and enabled.

Resolve the error

To resolve this error, you can either:

- Target a newer version of the .NET Framework.
- Enable the .NET Framework 3.5 on Windows 8.1 and then install the F# 2.0 runtime by repairing the Visual Studio installation. The steps to do this follow.

To enable the .NET Framework 3.5 on Windows 8.1

1. On the **Start** screen, type **Control Panel**.

As you type, the **Control Panel** icon appears under the **Apps** heading.

2. Choose the **Control Panel** icon, choose the **Programs** icon, and then choose the **Turn Windows features on or off** link.
3. Make sure that the **.NET Framework 3.5 (includes .NET 2.0 and 3.0)** check box is selected, and then choose the **OK** button. You don't need to select the check boxes for any child nodes for optional components of the .NET Framework.

The .NET Framework 3.5 is enabled if it wasn't already.

To install the F# 2.0 runtime

Follow the [steps to repair Visual Studio](#).

See also

- [F# guide \(.NET Framework\)](#)
- [F# in Visual Studio](#)

Work with 3D assets for games and apps

10/24/2019 • 3 minutes to read • [Edit Online](#)

This article describes the Visual Studio tools that you can use to create or modify 3D models, textures, and shaders for DirectX-based games and apps.

DirectX app development in Visual Studio

A DirectX app typically combines programming logic, the DirectX API, and High Level Shading Language (HLSL) programs, together with audio and 3D visual assets to present a rich, interactive multimedia experience. Visual Studio includes tools that you can use to work with images and textures, 3D models, and shaders without leaving the IDE to use another tool. The Visual Studio tools are especially suited for creating *placeholder* assets, which you can use to test code or build prototypes before you commission production-ready assets, and for inspecting and modifying production-ready assets when you are debugging your app.

Here's more information about the kinds of assets you can work with in Visual Studio.

Images and textures

Images and textures provide color and visual detail in games and apps. In 3D graphics, textures come in a variety of formats, types, and geometries to support different uses. For example, normal maps provide per-pixel surface normals for more-detailed lighting of 3D models, and cube maps provide texture in all directions for uses such as sky-boxing, reflections, and spherical texture mapping. Textures can provide mipmaps to support efficient rendering at different levels of detail, and can support different color channels and color orderings. Textures can be stored in a variety of compressed formats that occupy less dedicated graphics memory and help GPUs access textures more efficiently.

You can use the Visual Studio Image Editor to work with images and textures in many common types and formats.

3D models

3D models create space and shape in games and apps. Minimally, models encode the position of points in 3D space—which are known as *vertices*—together with indexing data to define lines or triangles that represent the shape of the model. Additional data can be associated with these vertices—for example, color information, normal vectors, or application-specific attributes. Each model can also define object-wide attributes—for example, which shader is used to compute the appearance of the object's surface, or which texture is applied to it.

You can use the Visual Studio Model Editor to work with 3D models in several common formats.

Shaders

Shaders are small, domain-specific programs that run on the graphics processing unit (GPU). Shaders determine how 3D models are transformed into on-screen shapes and how each pixel in those shapes is colored. By creating a shader and applying it to an object in your game or app, you can give the object a unique appearance.

You can use the Visual Studio Shader Designer, which is a graph-based shader design tool, to create custom visual effects without knowing HLSL programming.

NOTE

For more information about how to start with DirectX programming, see [DirectX](#). For more information about how to debug a DirectX-based app, see [Graphics diagnostics \(debugging DirectX graphics\)](#).

DirectX version compatibility

Visual Studio uses DirectX to render 2D and 3D assets. You can select either the DirectX 11 renderer, or the Windows Advanced Rasterization Platform (WARP) software renderer. The DirectX 11 renderer provides high-performance, hardware-accelerated rendering on DirectX 11 and DirectX 10 GPUs. The WARP renderer helps make sure that your assets work with a broad range of computers—this includes computers that don't have modern graphics hardware and computers that have integrated graphics hardware. For more information about WARP, see [Windows Advanced Rasterization Platform \(WARP\) guide](#).

Related topics

TITLE	DESCRIPTION
Working with textures and images	Describes how to use Visual Studio to work with images and textures.
Working with 3D models	Describes how to use Visual Studio to work with 3D models.
Working with shaders	Describes how to use the Visual Studio Shader Designer to create and modify custom shader effects.
Using 3D assets in your game or app	Describes how to use assets, which you created by using the Image Editor, Model Editor, or Shader Designer, in your game or app.

How to: Use 3D assets in your game or app

10/24/2019 • 8 minutes to read • [Edit Online](#)

This article describes how you can use Visual Studio to process 3D assets and include them in your builds.

After you use the tools in Visual Studio to create 3D assets, the next step is to use them in your app. But, before you can use them, your assets have to be transformed into a format that DirectX can understand. To help you transform your assets, Visual Studio provides build customizations for each kind of asset that it can produce. To include the assets in your build, all you have to do is configure your project to use the build customizations, add the assets to your project, and configure the assets to use the correct build customization. After that, you can load the assets into your app and use them by creating and filling DirectX resources just like you would in any other DirectX app.

Configure your project

Before you can deploy your 3D assets as part of your build, Visual Studio has to know about the kinds of assets that you want to deploy. Visual Studio already knows about many common file types, but because only certain kinds of apps use 3D assets, Visual Studio doesn't assume that a project will build these kinds of files. You can tell Visual Studio that your app uses these kinds of assets by using the *build customizations*—files that tell Visual Studio how to process different types of files in a useful way—that are provided for each asset type. Because these customizations are applied on a per-project basis, all you have to do is add the appropriate customizations to your project.

To add the build customizations to your project

1. In **Solution Explorer**, open the shortcut menu for the project, and then choose **Build Dependencies** > **Build Customizations**.

The **Visual C++ Build Customizations Files** dialog box appears.

2. Under **Available Build Customization Files**, select the check boxes that correspond to the asset types that you want to use in your project, as described in the following table:

ASSET TYPE	BUILD CUSTOMIZATION NAME
Textures and images	ImageContentTask(.targets, .props)
3D Models	MeshContentTask(.targets, .props)
Shaders	ShaderGraphContentTask(.targets, .props)

3. Choose the **OK** button.

Include assets in your build

Now that your project knows about the different kinds of 3D assets that you want to use, the next step is to tell it which files are 3D assets and what kinds of assets they are.

To add an asset to your build

1. In **Solution Explorer**, in your project, open the shortcut menu of an asset, and then choose **Properties**.

The asset's **Property Page** dialog box appears.

2. Make sure that the **Configuration** and **Platform** properties are set to the values to which you want your changes to apply.
3. Under **Configuration Properties**, choose **General**, and then in the property grid, under **General**, set the **Item Type** property to the appropriate content pipeline item type. For example, for an image or texture file, choose **Image Content Pipeline**.

IMPORTANT

By default, Visual Studio assumes that many kinds of image files should be categorized by using the **Image** item type that's built into Visual Studio. Therefore, you have to change the **Item Type** property of each image that you want to be processed by the image content pipeline. Other types of content pipeline source files for 3D models and visual shader graphics default to the correct **Item Type**.

4. Choose the **OK** button.

Following are the three content pipeline item types and their associated source and output file types.

ITEM TYPE	SOURCE FILE TYPES	OUTPUT FILE FORMAT
Image Content Pipeline	Portable Network Graphics (.png) JPEG (.jpg, .jpeg, .jpe, .jfif) Direct Draw Surface (.dds) Graphics Interchange Format (.gif) Bitmap (.bmp, .dib) Tagged Image File Format (.tif, .tiff) Targa (.tga)	DirectDraw Surface (.dds)
Mesh Content Pipeline	AutoDesk FBX Interchange File (.fbx) Collada DAE File (.dae) Wavefront OBJ File (.obj)	3D mesh file (.cmo)
Shader Content Pipeline	Visual Shader Graph (.dgsf)	Compiled Shader Output (.cso)

Configure asset content pipeline properties

You can set the content pipeline properties of each asset file so that it will be built in a specific way.

To configure content pipeline properties

1. In **Solution Explorer**, in your project, open the shortcut menu for the asset file, and then choose **Properties**.

The asset's **Property Page** dialog box appears.

2. Make sure that the **Configuration** and **Platform** properties are set to the values that you want your changes to apply to.
3. Under **Configuration Properties**, choose the content pipeline node (for example, **Image Content Pipeline** for texture and image assets), and then in the property grid, set the properties to the appropriate values. For example, to generate mipmaps for a texture asset at build time, set the **Generate Mips** property

to **Yes**.

4. Choose the **OK** button.

Image content pipeline configuration

When you use the image content pipeline tool to build a texture asset, you can compress the texture in various ways, indicate whether MIP levels should be generated at build time, and change the name of the output file.

PROPERTY	DESCRIPTION
Compress	<p>Specifies the compression type that's used for the output file.</p> <p>The available options are:</p> <ul style="list-style-type: none">- No Compression- BC1_UNORM compression- BC1_UNORM_SRGB compression- BC2_UNORM compression- BC2_UNORM_SRGB compression- BC3_UNORM compression- BC3_UNORM_SRGB compression- BC4_UNORM compression- BC4_SNORM compression- BC5_UNORM compression- BC5_SNORM compression- BC6H_UF16 compression- BC6H_SF16 compression- BC7_UNORM compression- BC7_UNORM_SRGB compression <p>For information about which compression formats are supported in different versions of DirectX, see Programming Guide for DXGI.</p>
Convert to pre-multiplied alpha format	<p>Yes to convert the image to pre-multiplied alpha format in the output file; otherwise, No. Only the output file is changed, the source image is unchanged.</p>
Generate Mips	<p>Yes to generate a full MIP chain at build time and include it in the output file; otherwise, No. If No, and the source file already contains a mipmap chain, then the output file will have a MIP chain; otherwise, the output file will have no MIP chain.</p>
Content Output	<p>Specifies the name of the output file. Important: Changing the file name extension of the output file has no effect on its file format.</p>

Mesh content pipeline configuration

When you use the mesh content pipeline tool to build a mesh asset, you can change the name of the output file.

PROPERTY	DESCRIPTION
Content Output	<p>Specifies the name of the output file. Important: Changing the file name extension of the output file has no effect on its file format.</p>

Shader content pipeline configuration

When you use the shader content pipeline tool to build a shader asset, you can change the name of the output file.

PROPERTY	DESCRIPTION
Content Output	Specifies the name of the output file. Important: Changing the file name extension of the output file has no effect on its file format.

Load and use 3D assets at run time

Use textures and images

Direct3D provides functions for creating texture resources. In Direct3D 11, the D3DX11 utility library provides additional functions for creating texture resources and resource views directly from image files. For more information about how to create a texture resource in Direct3D 11, see [Textures](#). For more information about how to use the D3DX11 library to create a texture resource or resource view from an image file, see [How to: Initialize a texture from a file](#).

Use 3D models

Direct3D 11 does not provide functions for creating resources from 3D models. Instead, you have to write code that reads the 3D model file and creates vertex and index buffers that represent the 3D model and any resources that the model requires—for example, textures or shaders.

Use shaders

Direct3D provides functions for creating shader resources and binding them to the programmable graphics pipeline. For more information about how to create a shader resource in Direct3D and bind it to the pipeline, see [Programming guide for HLSL](#).

In the programmable graphics pipeline, each stage of the pipeline must give the next stage of the pipeline a result that's formatted in a way that it can understand. Because the Shader Designer can only create pixel shaders, this means that it's up to your app to ensure that the data that it receives is in the format that it expects. Several programmable shader stages occur before the pixel shader and perform geometric transformations—the vertex shader, the hull shader, the domain shader, and the geometry shader. The non-programmable tessellation stage also occurs before the pixel shader. No matter which of these stages directly precedes the pixel shader, it must give its result in this format:

```
struct PixelShaderInput
{
    float4 pos : SV_POSITION;
    float4 diffuse : COLOR;
    float2 uv : TEXCOORD0;
    float3 worldNorm : TEXCOORD1;
    float3 worldPos : TEXCOORD2;
    float3 toEye : TEXCOORD3;
    float4 tangent : TEXCOORD4;
    float3 normal : TEXCOORD5;
};
```

Depending on the Shader Designer nodes that you use in your shader, you might also have to provide additional data in the format according to these definitions:

```

Texture2D Texture1 : register( t0 );
Texture2D Texture2 : register( t1 );
Texture2D Texture3 : register( t2 );
Texture2D Texture4 : register( t3 );
Texture2D Texture5 : register( t4 );
Texture2D Texture6 : register( t5 );
Texture2D Texture7 : register( t6 );
Texture2D Texture8 : register( t7 );

TextureCube CubeTexture1 : register( t8 );
TextureCube CubeTexture2 : register( t9 );
TextureCube CubeTexture3 : register( t10 );
TextureCube CubeTexture4 : register( t11 );
TextureCube CubeTexture5 : register( t12 );
TextureCube CubeTexture6 : register( t13 );
TextureCube CubeTexture7 : register( t14 );
TextureCube CubeTexture8 : register( t15 );

SamplerState TexSampler : register( s0 );

cbuffer MaterialVars : register (b0)
{
    float4 MaterialAmbient;
    float4 MaterialDiffuse;
    float4 MaterialSpecular;
    float4 MaterialEmissive;
    float MaterialSpecularPower;
};

cbuffer LightVars : register (b1)
{
    float4 AmbientLight;
    float4 LightColor[4];
    float4 LightAttenuation[4];
    float3 LightDirection[4];
    float LightSpecularIntensity[4];
    uint IsPointLight[4];
    uint ActiveLights;
};

cbuffer ObjectVars : register(b2)
{
    float4x4 LocalToWorld4x4;
    float4x4 LocalToProjected4x4;
    float4x4 WorldToLocal4x4;
    float4x4 WorldToView4x4;
    float4x4 UVTransform4x4;
    float3 EyePosition;
};

cbuffer MiscVars : register(b3)
{
    float ViewportWidth;
    float ViewportHeight;
    float Time;
};

```

Related topics

TITLE	DESCRIPTION
How to: Export a texture that contains mipmaps	Describes how to use the Image Content Pipeline to export a texture that contains precomputed mipmaps.

TITLE	DESCRIPTION
How to: Export a texture that has premultiplied alpha	Describes how to use the Image Content Pipeline to export a texture that contains premultiplied alpha values.
How to: Export a texture for use with Direct2D or JavaScript apps	Describes how to use the Image Content Pipeline to export a texture that can be used in a Direct2D or JavaScript app.
Working with 3D assets for games and apps	Describes the editing tools that Visual Studio provides for creating and manipulating 3D assets, which include textures and images, 3D models, and shaders.
How to: Export a shader	Describes how to export a shader from the Shader Designer.

Work with textures and images

10/18/2019 • 2 minutes to read • [Edit Online](#)

You can use the Image Editor in Visual Studio to create and modify textures and images. The Image Editor supports rich texture and image formats like those that are used in DirectX app development.

NOTE

The Image Editor doesn't support low-color images like icons or cursors. To create or modify those kinds of images, use the [Image Editor for icons \(C++\)](#).

Textures and images

Textures and images are, at a basic level, just tables of data that are used to provide visual detail in graphics apps. The kind of detail that a texture or image provides depends on how it's used, but color samples, alpha (transparency) values, surface normals, and height values are common examples. The primary difference between a texture and an image is that a texture is meant to be used together with a representation of shape—typically a 3D model—to express a complete object or scene, but an image is typically a stand-alone representation of the object or scene.

Any texture can be encoded and compressed in a number of ways that are orthogonal to the type of data that a texture holds, or to the dimensionality or "shape" of the texture. However, different encoding and compression methods yield better results for different kinds of data.

You can use the Image Editor to create and modify textures and images in ways that resemble other image editors. The Image Editor also provides mipmapping and other features for use with 3D graphics, and supports many of the highly-compressed, hardware-accelerated texture formats that DirectX supports.

Common kinds of textures include:

Texture maps

Texture maps contain color values that are organized as a one-, two-, or three-dimensional matrix. They are used to provide color detail on the affected object. Colors are commonly encoded by using RGB (red, green, blue) color channels, and may include a fourth channel, alpha, that represents transparency. Less commonly, colors could be encoded in another color scheme, or the fourth channel could contain data other than alpha—for example, height.

Normal maps

Normal maps contain surface normals. They are used to provide lighting detail on the affected object. Normals are commonly encoded by using the red, green, and blue color components to store the x, y, and z dimensions of the vector. However, other encodings exist—for example, encodings that are based on polar coordinates.

Height maps

Height maps contain height-field data. They are used to provide a form of geometric detail on the affected object—by using shader code to compute the desired effect—or to provide data points for uses like terrain generation. Height values are commonly encoded by using one channel in a texture.

Cube maps

Cube maps can contain different types of data—for example, colors or normals—but are organized as six textures on the faces of a cube. Because of this, cube maps are not sampled by supplying texture coordinates, but by supplying a vector whose origin is the center of the cube; the sample is taken at the point where the vector

intersects the cube. Cube maps are used to provide an approximation of the environment that can be used to calculate reflections—this is known as *environment mapping*—or to provide texture to spherical objects with less distortion than basic, two-dimensional textures can provide.

Related topics

TITLE	DESCRIPTION
Image Editor	Describes how to use the Image Editor to work with textures and images.
Image Editor examples	Provides links to topics that demonstrate how to use the Image Editor to perform common image processing tasks.

Image editor

10/18/2019 • 18 minutes to read • [Edit Online](#)

This article describes how to work with the Visual Studio **Image Editor** to view and modify texture and image resources.

You can use the **Image Editor** to work with the kinds of rich texture and image formats that are used in DirectX app development. This includes support for popular image file formats and color encodings, features such as alpha-channels and MIP-mapping, and many of the highly compressed, hardware-accelerated texture formats that DirectX supports.

Supported formats

The **Image Editor** supports the following image formats:

FORMAT NAME	FILE NAME EXTENSION
Portable Network Graphics	.png
JPEG	.jpg, .jpeg, .jpe, .jfif
Direct Draw Surface	.dds
Graphics Interchange Format	.gif
Bitmap	.bmp, .dib
Tagged Image File Format	.tif, .tiff
TGA (Targa)	.tga

Get started

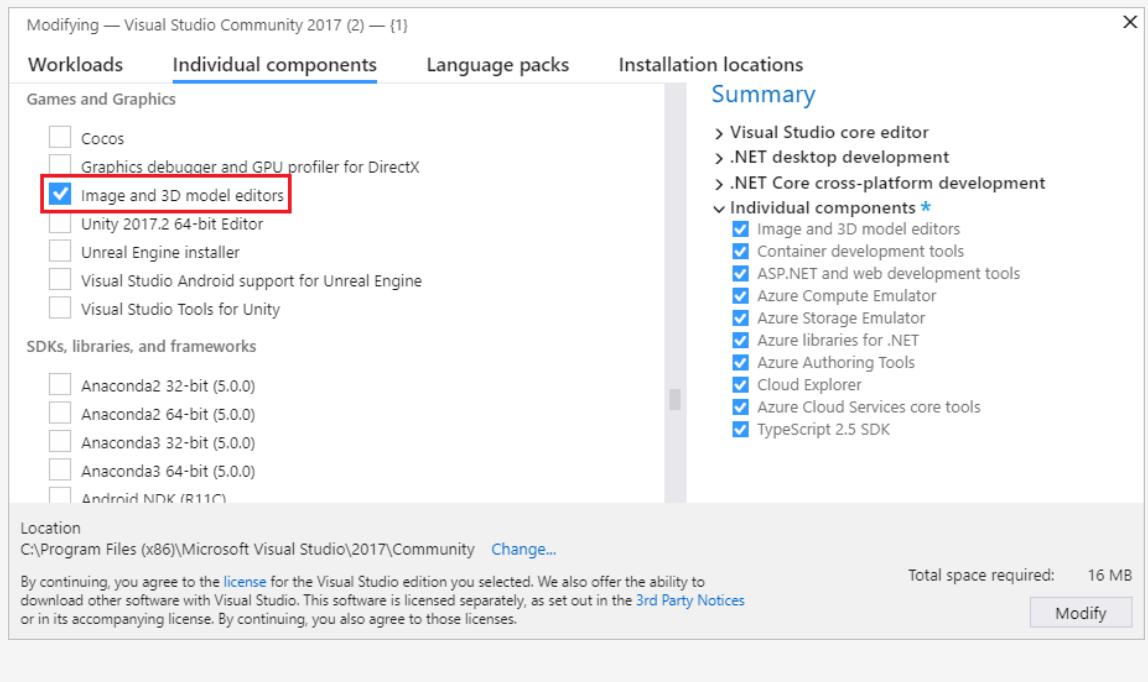
This section describes how to add an image to your Visual Studio project and configure it for your requirements.

Add an image to your project

1. In **Solution Explorer**, open the shortcut menu for the project that you want to add the image to, and then choose **Add > New Item**.
2. In the **Add New Item** dialog box, under **Installed**, select **Graphics**, and then select an appropriate file format for the image.

NOTE

If you don't see the **Graphics** category in the **Add New Item** dialog, you may need to install the **Image and 3D model editors** component. Close the dialog and then select **Tools > Get Tools and Features** from the menu bar, to open the **Visual Studio Installer**. Select the **Individual components** tab, and then select the **Image and 3D model editors** component under the **Games and Graphics** category. Select **Modify**.



For information about how to choose a file format based on your requirements, see [Choose the image format](#).

3. Specify the **Name** of the image file and the **Location** where you want it to be created.

4. Choose the **Add** button.

Choose the image format

Depending on how you plan to use the image, certain file formats might be more appropriate than others. For example, some formats might not support a feature that you need, for example, transparency or a specific color format. Some formats might not provide suitable compression for the kind of image content you have planned.

The following information can help you choose an image format that meets your needs:

Bitmap Image (.bmp)

The bitmap image format. An uncompressed image format that supports 24-bit color. The bitmap format doesn't support transparency.

GIF Image (.gif)

The Graphics Interchange Format (GIF) image format. An LZW-compressed, lossless image format that supports up to 256 colors. Unsuitable for photographs and images that have a significant amount of color detail, but provides good compression ratios for low-color images that have a high degree of color coherence.

JPG Image (.jpg)

The Joint Photographic Experts Group (JPEG) image format. A highly compressed, lossy image format that supports 24-bit color and is suitable for general-purpose compression of images that have a high degree of color coherence.

PNG Image (.png)

The Portable Network Graphics (PNG) image format. A moderately compressed, lossless image format that supports 24-bit color and alpha transparency. It is suitable for both natural and artificial images, but does not provide compression ratios as good as lossy formats such as JPG or GIF.

TIFF Image (.tif)

The Tagged Image File Format (TIFF or TIF) image format. A flexible image format that supports several compression schemes.

DDS Texture (.dds)

The DirectDraw Surface (DDS) texture format. A highly compressed, lossy texture format that supports 24-bit color and alpha transparency. Its compression ratios can be as high as 8:1. It's based on S3 Texture compression, which can be decompressed on graphics hardware.

TGA Image (.tga)

The Truevision Graphics Adapter (TGA) image format (also known as Targa). An RLE-compressed, lossless image format that supports both color-mapped (color palette) or direct-color images of up to 24-bit color and alpha transparency. Unsuitable for photographs and images that have a significant amount of color detail, but provides good compression ratios for images that have long spans of identical colors.

Configure the image

Before you begin to work with the image that you created, you can change its default configuration. For example, you can change its dimensions or the color format that it uses. For information about how to configure these and other properties of the image, see [Image properties](#).

NOTE

Before you save your work, make sure to set the **Color Format** property if you want to use a specific color format. If the file format supports compression, you can adjust the compression settings when you save the file for the first time or when you choose **Save As**.

Work with the Image Editor

This section describes how to use the **Image Editor** to modify textures and images.

Commands that affect the state of the **Image Editor** are located on the **Image Editor Mode** toolbar together with advanced commands. The toolbar is located along the topmost edge of the **Image Editor** design surface. Drawing tools are located on the **Image Editor** toolbar along the leftmost edge of the **Image Editor** design surface.

Image Editor Mode toolbar



The following table describes the items on the **Image Editor Mode** toolbar, which are listed in the order in which they appear from left to right:

TOOLBAR ITEM	DESCRIPTION
Select	Enables selection of a rectangular region of an image. After you select a region, you can cut, copy, move, scale, rotate, flip, or delete it. When there is an active selection, drawing tools only affect the selected region.

TOOLBAR ITEM	DESCRIPTION
Irregular Selection	Enables selection of an irregular region of an image. After you select a region, you can cut, copy, move, scale, rotate, flip, or delete it. When there is an active selection, drawing tools only affect the selected region.
Wand Selection	Enables selection of a similarly colored region of an image. The <i>tolerance</i> —that is, the maximum difference between adjacent colors within which they are considered similar—can be configured to include a smaller or wider range of similar colors. After you select a region, you can cut, copy, move, scale, rotate, flip, or delete it. When there is an active selection, drawing tools only affect the selected region.
Pan	<p>Enables movement of the image relative to the window frame. In Pan mode, select a point on the image and then move it around.</p> <p>You can temporarily activate Pan mode by pressing and holding the Ctrl key.</p>
Zoom	<p>Enables the display of more or less image detail relative to the window frame. In Zoom mode, select a point on the image and then move it right or down to zoom in, or left or up to zoom out.</p> <p>You can zoom in or out by pressing and holding Ctrl while you either use the mouse wheel or press the plus sign (+) or minus sign (-).</p>
Zoom to Actual Size	Displays the image by using a 1:1 relationship between the pixels of the image and the pixels of the screen.
Zoom To Fit	Displays the full image in the window frame.
Zoom To Width	Displays the full width of the image in the window frame.
Grid	Enables or disables the grid that shows pixel boundaries. The grid might not appear until you zoom into the image.
View Next MIP Level	Activates the next larger MIP level in a MIP map chain. The active MIP level is displayed on the design surface. This item is only available for textures that have MIP levels.
View Previous MIP Level	Activates the next smaller MIP level in a MIP map chain. The active MIP level is displayed on the design surface. This item is only available for textures that have MIP levels.
Red Channel	Enables or disables the specific color channel. Note: By systematically enabling or disabling color channels, you can isolate problems that are related to one or more of them. For example, you could identify incorrect alpha transparency.
Green Channel	
Blue Channel	
Alpha Channel	

TOOLBAR ITEM	DESCRIPTION
Background	<p>Enables or disables display of the background through transparent parts of the image. You can configure how the background is displayed by choosing from these options:</p> <p>Checkerboard Uses a green color together with the specified background color to display the background as a checkerboard pattern. You can use this option to help make transparent parts of the image more apparent.</p> <p>White Background Uses the color white to display the background.</p> <p>Black Background Uses the color black to display the background.</p> <p>Animate Background Pans the checkerboard pattern slowly. You can use this option to help make transparent parts of the image more apparent.</p>
Properties	Alternately opens or closes the Properties window.

TOOLBAR ITEM	DESCRIPTION
Advanced	<p>Contains additional commands and options.</p> <p>Filters</p> <p>Provides several common image filters: Black and White, Blur, Brighten, Darken, Edge Detection, Emboss, Invert Colors, Ripple, Sepia Tone, and Sharpen.</p> <p>Graphics Engines</p> <p>Render with D3D11 Uses Direct3D 11 to render the Image Editor design surface.</p> <p>Render with D3D11WARP Uses Direct3D 11 Windows Advanced Rasterization Platform (WARP) to render the Image Editor design surface.</p> <p>Tools</p> <p>Flip Horizontal Transposes the image around its horizontal, or x, axis.</p> <p>Flip Vertical Transposes the image around its vertical, or y, axis.</p> <p>Generate Mips Generates MIP levels for an image. If MIP levels already exist, they are recreated from the largest MIP level. Any changes that were made to smaller MIP levels are lost. To save the MIP levels that you have generated, you must use the .dds format to save the image.</p> <p>View</p> <p>Frame Rate When enabled, displays the frame rate in the upper-right corner of the design surface. The frame rate is the number of frames that are drawn per second. Tip: You can choose the Advanced button to run the last command again.</p>

Image Editor toolbar



The following table describes the items on the **Image Editor** toolbar, which are listed in the order in which they appear from top to bottom:

Toolbar Item	Description
Pencil	Uses the active color selection to draw an aliased stroke. You can set the color and thickness of the stroke in the Properties window.
Brush	Uses the active color selection to draw an anti-aliased stroke. You can set the color and thickness of the stroke in the Properties window.
Airbrush	Uses the active color selection to draw an anti-aliased stroke that blends together with the image and becomes more saturated as a function of time. You can set the color and thickness of the stroke in the Properties window.
Eyedropper	Sets the active color selection to the color of the selected pixel.
Fill	<p>Uses the active color selection to fill a region of the image. The affected region is defined as the pixel where the fill is applied, together with every pixel that is connected to it by pixels of the same color and that is the same color itself. If the fill is applied within an active selection, then the affected region is constrained by the selection.</p> <p>By default, the active color selection is blended together with the affected region of the image according to its alpha component. To use the active color selection to overwrite the affected region, press and hold the Shift key when you use the fill tool.</p>
Eraser	Sets pixels to the fully transparent color if the image supports an alpha channel. Otherwise, sets the pixels to the active background color.
Line, Rectangle, Rounded Rectangle, Ellipse	<p>Draws a shape on the image. You can set the color and thickness of the outline in the Properties window.</p> <p>To draw a primitive that has equal width and height, press and hold Shift as you draw.</p>
Text	Uses the foreground color selection to draw text. The background color is determined by the background color selection. For a transparent background, the alpha value of the background color selection must be 0. While the text region is active, you can set whether the text is drawn with an anti-aliased stroke, and you can set the text Value , Font , Size , and style— Bold , Italics , or Underlined —in the Properties window. The content and appearance of the text is finalized when the text region is no longer active.
Rotate	Rotates the image 90 degrees clockwise.
Trim	Trims the image to the active selection.

Work with MIP levels

Some image formats, for example, DirectDraw Surface (*.dds*), support MIP levels for texture-space Level-of-Detail (LOD). For information about how to generate and work with MIP levels, see [How to: Create and modify](#)

MIP levels

Work with transparency

Some image formats, for example, DirectDraw Surface (.dds), support transparency. There are several ways you can use transparency, depending on the tool that you're using. To specify the level of transparency for a color selection, in the **Properties** window, set the **A** (alpha) component of the color selection.

The following table describes how different kinds of tools control how transparency is applied:

TOOL	DESCRIPTION
Pencil, Brush, Airbrush, Line, Rectangle, Rounded Rectangle, Ellipse, Text	To blend the active color selection together with the image, in the Properties window, expand the Channels property group and set the Draw checkbox on the Alpha channel, and then draw normally.
	To draw by using the active color selection and leave the alpha value of the image in place, clear the Draw checkbox of the Alpha channel, and then draw normally.
Fill	To blend the active color selection together with the image, just choose the area to fill.
	To use the active color selection—including the value of the alpha channel—to overwrite the image, press and hold Shift and then choose the area to fill.

Image properties

You can use the **Properties** window to specify various properties of the image. For example, you can set the width and height properties to resize the image.

The following table describes image properties:

PROPERTY	DESCRIPTION
Width	The width of the image.
Height	The height of the image.
Bits Per Pixel	The number of bits that represent each pixel. The value of this property depends on the Color Format of the image.
Transparent Selection	True to blend the selection layer together with the main image, based on the alpha value of the selection layer; otherwise, False . This item is only available for images that support alpha.
Format	The color format of the image. You can specify a variety of color formats, depending on the image format. The color format defines the number and kind of color channels that are included in the image, and also the size and encoding of various channels.
Mip Level	The active MIP level. This item is only available for textures that have MIP levels.

PROPERTY	DESCRIPTION
Mip Level Count	The total number of MIP levels in the image. This item is only available for textures that have MIP levels.
Frame Count	The total number of frames in the image. This item is only available for images that support texture arrays.
Frame	The current frame. Only the first frame can be viewed; all other frames are lost when the image is saved.
Depth Slice Count	The total number of depth slices in the image. This item is only available for images that support volume textures.
Depth Slice	The current depth slice. Only the first slice can be viewed; all other slices are lost when you save the image.

NOTE

Because the **Rotate by** property applies to all tools and selected regions, it always appears at the bottom of the **Properties** window together with other tool properties. **Rotate by** is always displayed because the whole image is implicitly selected when there is no other selection or active tool. For more information about the **Rotate by** property, see [Tool properties](#tool -properties).

Resize images

There are two ways to resize an image. In both cases, the **Image Editor** uses bilinear interpolation to resample the image.

- In the **Properties** window, specify new values for the **Width** and **Height** properties.
- Select the entire image and use the border markers to resize the image.

Selected regions

Selections in the **Image Editor** define regions of the image that are active. Active regions are affected by tools and transformations. When there is an active selection, areas outside the selected region are not affected by most tools and transformations. If there is not an active selection, the entire image is active.

Most tools (**Pencil**, **Brush**, **Airbrush**, **Fill**, **Eraser**, and 2D primitives) and transformations (**Rotate**, **Trim**, **Invert Colors**, **Flip Horizontal**, and **Flip Vertical**) are constrained or defined by the active selection. However, some tools (**Eyedropper** and **Text**) and transformations (**Generate Mips**) aren't affected by any active selection. These tools always behave as if the entire image is the active selection.

While you're selecting a region, you can press and hold **Shift** to make a proportional (square) selection. Otherwise, the selection is not constrained.

Resize selections

After you select a region, you can resize it or its image contents by changing the size of the selection marker. While you're resizing the selected region, you can use the following modifier keys to change the behavior of the selected region as you resize it:

Ctrl - Copies the contents of the selected region before it's resized. This leaves the original image intact while the copy is resized.

Shift - Resizes the selected region in proportion to its original size.

Alt - Changes the size of the selection region. This leaves the image unmodified.

The following table describes the valid modifier key combinations:

CTRL	SHIFT	ALT	DESCRIPTION
			Resizes the content of the selected region.
	Shift		Proportionally resizes the content of the selected region.
		Alt	Resizes the selected region. This defines a new selection region.
	Shift	Alt	Proportionally resizes the selected region. This defines a new selection region.
Ctrl			Copies and then resizes the content of the selected region.
Ctrl	Shift		Copies and then proportionally resizes the content of the selected region.

Tool properties

While a tool is selected, you can use the **Properties** window to specify details about how it affects the image. For example, you can set the thickness of the **Pencil** tool or the color of the **Brush** tool.

You can set both a foreground color and a background color. Both support an alpha channel to provide user-defined opacity. The settings apply to all tools. If you use a mouse, the left mouse button corresponds to the foreground color, and the right mouse button corresponds to the background color.

The following table describes tool properties:

TOOL	PROPERTIES
All tools and selections	Rotate by Defines the amount, in degrees, that the selection or tool effect is rotated in the clockwise direction.
Pencil, Brush, Airbrush, Eraser	Thickness Defines the size of the area that is affected by the tool.

Tool	Properties
Text	<p>Anti-alias Draws text that has anti-aliased edges. This gives text a smoother appearance.</p> <p>Value The text to be drawn.</p> <p>Font The font used to draw the text.</p> <p>Size The size of the text.</p> <p>Bold Makes the font bold.</p> <p>Italics Makes the font italic.</p> <p>Underlined Makes the font underlined.</p>
2D Primitive	<p>Anti-alias Draws primitives that have anti-aliased edges. This gives them a smoother appearance.</p> <p>Thickness Defines the thickness of the line that forms the boundary of the primitive.</p> <p>Radius X (Rounded rectangle only) Defines the rounding radius for the top and bottom edges of the primitive.</p> <p>Radius Y (Rounded rectangle only) Defines the rounding radius for the left and right edges of the primitive.</p>
Pencil, Brush, Airbrush, 2D Primitive	<p>Channels Enables or disables specific color channels for viewing and drawing. If View is set for a specific color channel, that channel is visible in the image; otherwise, it is not visible. If Draw is set for a specific color channel, that channel is affected by drawing operations; otherwise, it is not.</p>
Wand Selection, Fill	<p>Tolerance Defines the maximum difference between adjacent colors within which they are considered similar, so that fewer or more similar colors are made a part of the affected or selected region. By default, the value is 32, which means that adjacent pixels within 32 shades (lighter or darker) of the original color are considered to be part of the region.</p>

Keyboard shortcuts

Command	Keyboard Shortcuts
Switch to Select mode	S

COMMAND	KEYBOARD SHORTCUTS
Switch to Zoom mode	Z
Switch to Pan mode	K
Select all	Ctrl+A
Delete the current selection	Delete
Cancel the current selection	Esc (escape)
Zoom in	Ctrl+Mouse wheel forward Ctrl+PageUp Plus Sign (+)
Zoom out	Ctrl-Mouse wheel backward Ctrl-PageDown Minus Sign (-)
Pan the image up	Mouse wheel backward PageDown
Pan the image down	Mouse wheel forward PageUp
Pan the image left	Shift+Mouse wheel backward Mouse wheel left Shift+PageDown
Pan the image right	Shift+Mouse wheel forward Mouse wheel right Shift+PageUp
Zoom to actual size	Ctrl+0 (zero)
Fit image to window	Ctrl+G, Ctrl+F
Fit image to window width	Ctrl+G, Ctrl+I
Toggle grid	Ctrl+G, Ctrl+G
Crop image to current selection	Ctrl+G, Ctrl+C
View next (higher detail) MIP level	Ctrl+G, Ctrl+6

COMMAND	KEYBOARD SHORTCUTS
View previous (lower detail) MIP level	Ctrl+G, Ctrl+7
Toggle red color channel	Ctrl+G, Ctrl+1
Toggle green color channel	Ctrl+G, Ctrl+2
Toggle blue color channel	Ctrl+G, Ctrl+3
Toggle alpha (transparency) channel	Ctrl+G, Ctrl+4
Toggle alpha checkerboard pattern	Ctrl+G, Ctrl+B
Switch to irregular selection tool	L
Switch to wand selection tool	M
Switch to pencil tool	P
Switch to brush tool	B
Switch to fill tool	F
Switch to eraser tool	E
Switch to text tool	T
Switch to color-select (eyedropper) tool	I
Move the active selection, and its contents.	Arrow keys.
Resize the active selection, and its contents.	Ctrl+Arrow keys
Move the active selection, but not its contents.	Shift+Arrow keys
Resize the active selection, but not its contents.	Shift+Ctrl+Arrow keys
Commit the current layer	Return
Decrease tool thickness	[
Increase tool thickness]

Related topics

TITLE	DESCRIPTION
Working with 3D assets for games and apps	Provides an overview of the tools that you can use in Visual Studio to work with graphics assets such as textures and images, 3D models, and shader effects.

TITLE	DESCRIPTION
Model editor	Describes how to use the Visual Studio Model Editor to work with 3D models.
Shader designer	Describes how to use the Visual Studio Shader Designer to work with shaders.

How to: Create a basic texture

10/18/2019 • 3 minutes to read • [Edit Online](#)

This article demonstrates how to use the Image Editor to create a basic texture, including the following activities:

- Setting the size of the texture
- Setting the foreground and background colors
- Using the alpha channel (transparency)
- Using the **Fill** and **Ellipse** tools
- Setting tool properties

Create a basic texture

You can use the Image Editor to create and modify images and textures for your game or app.

The following steps show how to create a texture that represents a "bullseye" target. When you are finished, the texture should look like the following picture. To better demonstrate the transparency in the texture, the Image Editor has been configured to use a green, checkered pattern to display it.



Before you begin, make sure that the **Properties** window is displayed. You use the **Properties** window to set the image size, change tool properties, and specify colors while you work.

Create a "bullseye" target texture

1. Create a texture with which to work. For information about how to add a texture to your project, see [Image Editor](#).
2. Set the image size to 512x512 pixels. In the **Properties** window, set the values of the **Width** and **Height** properties to `512`.
3. On the Image Editor toolbar, choose the **Fill** tool. The **Properties** window now displays the properties of the **Fill** tool together with the image properties.
4. Set the foreground color to fully transparent black. In the **Properties** window, in the **Colors** property group, select **Foreground**. Set the values of the **R**, **G**, **B**, and **A** properties next to the color picker to `0`.
5. On the Image Editor toolbar, choose the **Fill** tool, and then press and hold the **Shift** key and choose any point in the image. Using the **Shift** key causes the alpha value of the fill color to replace the color in the

image; otherwise, the alpha value is used to blend the fill color together with the color in the image.

IMPORTANT

This step, together with the color selection in the previous step, ensures that the base image is prepared for the "bullseye" target texture that you will draw. When the image is filled with transparent black—and because the border of the target is black—there will be no aliasing artifacts around the target.

6. On the Image Editor toolbar, choose the **Ellipse** tool.
7. Set the foreground color to fully opaque black. Set the values of the **R**, **G**, and **B** properties to **0** and the value of the **A** property to **255**.
8. Set the background color to fully opaque white. In the **Properties** window, in the **Colors** property group, select **Background**. Set the values of the **R**, **G**, **B**, and **A** properties to **255**.
9. Set the width of the outline of the ellipse. In the **Properties** window, in the **Appearance** property group, set the value of the **Width** property to **8**.
10. Make sure that anti-aliasing is enabled. In the **Properties** window, in the **Appearance** property group, make sure that the **Anti-alias** property is set.
11. Using the **Ellipse** tool, draw a circle from pixel coordinate **(3, 3)** to pixel coordinate **(508, 508)**. To draw the circle more easily, you can press and hold the **Shift** key while you draw.

NOTE

The pixel coordinates of the current pointer location are displayed on the Visual Studio status bar.

12. Change the background color. Set **R** to **44**, **G** to **165**, **B** to **211**, and **A** to **255**.
13. Draw another circle from pixel coordinate **(64, 64)** to pixel coordinate **(448, 448)**.
14. Change the background color back to fully opaque white. Set **R**, **G**, **B**, and **A** to **255**.
15. Draw another circle from pixel coordinate **(128, 128)** to pixel coordinate **(384, 384)**.
16. Change the background color. Set **R** to **255**, **G** and **B** to **64**, and **A** to **255**.
17. Draw another circle from pixel coordinate **(192, 192)** to pixel coordinate **(320, 320)**.

The "bullseye" target texture is complete. Here's the final image, shown with transparency.



As a next step, you can generate MIP levels for this texture. For information, see [How to: Create and modify MIP](#)

levels.

See also

- [Image Editor](#)

How to: Create and modify MIP levels

10/18/2019 • 2 minutes to read • [Edit Online](#)

This document demonstrates how to use the **Image Editor** to generate and modify *MIP levels* for texture-space Level-of-Detail (LoD).

Generating MIP levels

Mipmapping is a technique that's used to increase rendering speed and reduce aliasing artifacts on textured objects by pre-calculating and storing several copies of a texture in different sizes. Each copy, which is known as a MIP level, is half the width and height of the previous copy. When a texture is rendered on the surface of an object, the MIP level that corresponds most closely to the screen-space area of the textured surface is automatically chosen. This means that the graphics hardware doesn't have to filter oversized textures to maintain consistent visual quality. Although the memory cost of storing the MIP levels is about 33 percent more than that of the original texture alone, the performance and image-quality gains justify it.

To generate MIP levels

1. Begin with a basic texture, as described in [How to: Create a basic texture](#). For best results, specify a texture that has a width and height that are a power of two in size, for example, 256, 512, 1024, and so on.
2. Generate the MIP levels. On the **Image Editor Mode** toolbar, choose **Advanced > Tools > Generate Mips**.

Notice that the **Go to Next Mip Level** and **Go to Previous Mip Level** buttons now appear on the **Image Editor Mode** toolbar. If the **Properties** window is displayed, also notice that the read-only properties **Mip Level** and **Mip Level Count** now appear in the image properties.

Modifying MIP levels

To achieve special effects or increase image quality at specific levels of detail, you can modify each MIP level individually. For example, you can give a textured object a different appearance at a distance (greater distance corresponds to smaller MIP levels), or you can ensure that textures that contain text or symbols remain legible even at smaller MIP levels.

To modify an individual MIP level

1. Select the MIP level that you want to modify. On the **Image Editor Mode** toolbar, use the **Go to Next MIP Level** and **Go to Previous MIP Level** buttons to move between MIP levels.
2. After you select the MIP level that you want to modify, you can use the drawing tools to modify it without changing the contents of other MIP levels. The drawing tools are available on the **Image Editor** toolbar. After you select a tool, you can change its properties in the **Properties** window. For information about the drawing tools and their properties, see [Image Editor](#).

NOTE

If you do not need to modify the contents of individual MIP levels—as you might do to achieve certain effects—we recommend that you generate mipmaps from the source texture at build time. This helps to ensure that MIP levels stay in sync with the source texture because modifications to a MIP level are not propagated to other levels automatically. For more information on how to generate mipmaps at build time, see [How to: Export a texture that contains mipmaps](#).

See also

- [How to: Create a basic texture](#)

Work with 3D models

10/18/2019 • 2 minutes to read • [Edit Online](#)

You can use the Model Editor in Visual Studio to create 3D models. You can use the models in your DirectX-based game or app.

3D models

3D models define the shape of objects as they exist in a 3D scene. Models can be basic solitary objects, complex objects that are formed from hierarchies of basic objects, or even entire 3D scenes. A 3D object is made up of points in 3D space (known as *vertices*), indices that define triangles, lines, or other primitives that are made up of those points, and attributes that might apply on a per-vertex or per-primitive basis—for example, surface normals. Additionally, some information might apply on a per-object basis—for example, which shader and textures will give the object its unique appearance.

The Model Editor is the only tool you need to create basic 3D models—complete with material properties, textures, and pixel shaders—that you can use in your game or app. Or, you can create placeholder models to use for prototyping and testing before you engage artists to finalize the models.

You can also use the Model Editor to view existing 3D models that have been created by using full-featured tools, and to modify them if you observe problems in the art assets.

Related topics

TITLE	DESCRIPTION
Model Editor	Describes how to use the Model Editor to work with 3D models.
Model Editor examples	Provides links to topics that demonstrate how to use the Model Editor to perform common 3D modeling tasks.

Model editor

10/18/2019 • 21 minutes to read • [Edit Online](#)

This document describes how to work with the Visual Studio **Model Editor** to view, create, and modify 3D models.

You can use **Model Editor** to create basic 3D models from scratch, or to view and modify more-complex 3D models that were created by using full-featured 3D modeling tools.

Supported formats

The **Model Editor** supports several 3D model formats that are used in DirectX app development:

FORMAT NAME	FILE EXTENSION	SUPPORTED OPERATIONS (VIEW, EDIT, CREATE)
AutoDesk FBX Interchange File	.fbx	View, Edit, Create
Collada DAE File	.dae	View, Edit (Modifications to Collada DAE files are saved by using the FBX format.)
OBJ	.obj	View, Edit (Modifications to OBJ files are saved by using the FBX format.)

Get started

This section describes how to add a 3D model to your Visual Studio C++ project and other basic information that will help you get started.

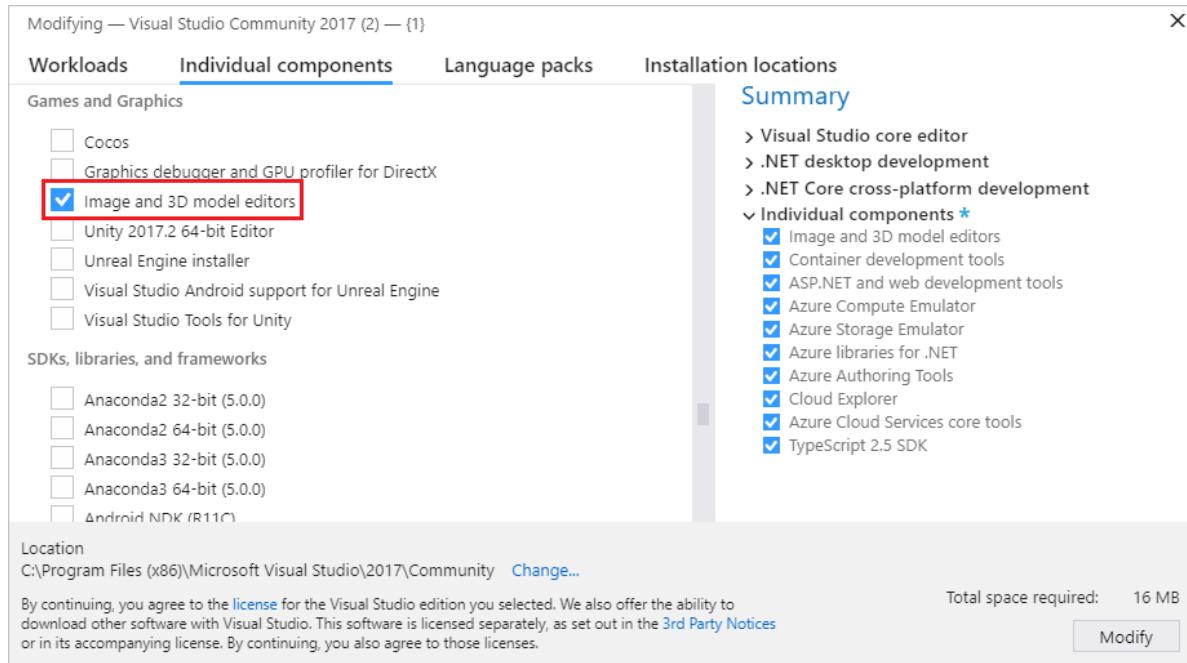
NOTE

Automatic build integration of graphics items like 3D scenes (.fbx files) is only supported for C++ projects.

To add a 3D model to your project

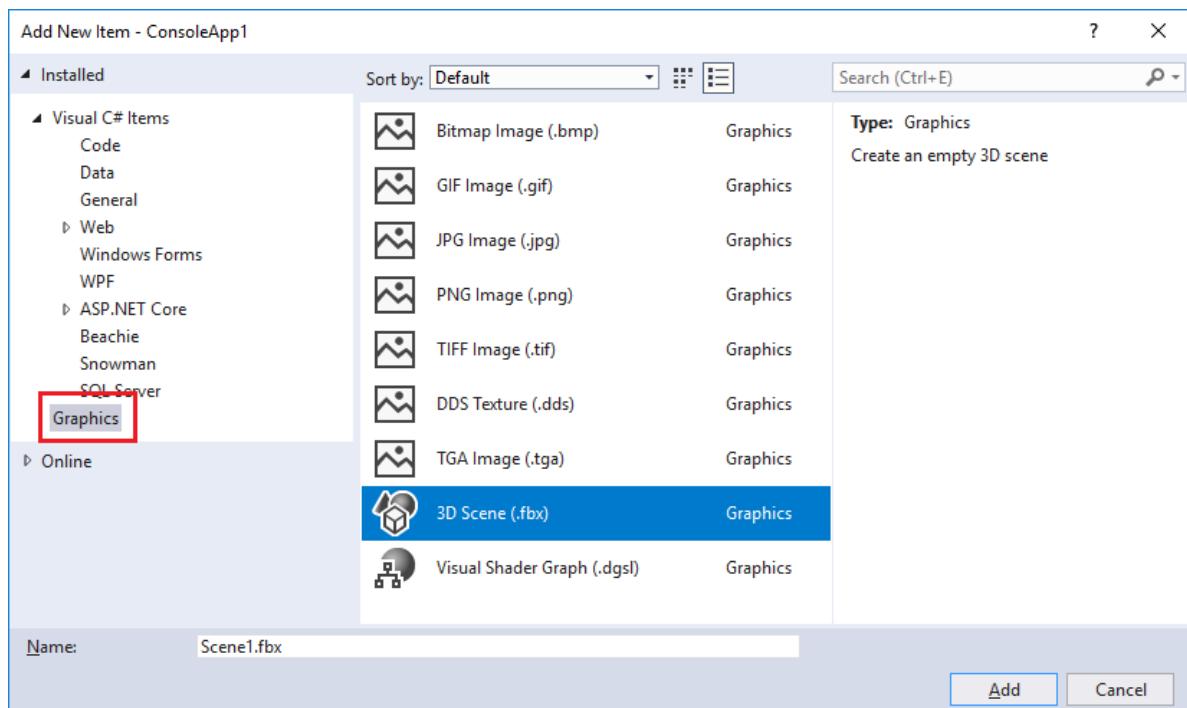
1. Ensure you have the required Visual Studio component installed that you need to work with graphics. The component is called **Image and 3D model editors**.

To install it, open Visual Studio Installer by selecting **Tools > Get Tools and Features** from the menu bar, and then select the **Individual components** tab. Select the **Image and 3D model editors** component under the **Games and Graphics** category, and then select **Modify**.



The component starts installing.

2. In **Solution Explorer**, open the shortcut menu for the C++ project you want to add the image to, and then choose **Add > New Item**.
3. In the **Add New Item** dialog box, under the **Graphics** category, select **3D Scene (.fbx)**.



NOTE

If you don't see the **Graphics** category in the **Add New Item** dialog, and you have the **Image and 3D model editors** component installed, graphics items are not supported for your project type.

4. Enter the **Name** of the model file, and then select **Add**.

Axis orientation

Visual Studio supports every orientation of the 3D axis, and loads axis orientation information from model file formats that support it. If no axis orientation is specified, Visual Studio uses the right-handed coordinate system

by default. The **axis indicator** shows the current axis orientation in the lower-right corner of the design surface. On the **axis indicator**, red represents the x-axis, green represents the y-axis, and blue represents the z-axis.

Begin your 3D model

In the Model Editor, each new object always begins as one of the basic 3D shapes—or *primitives*—that are built into the Model Editor. To create new and unique objects you add a primitive to the scene and then change its shape by modifying its vertices. For complex shapes, you add additional vertices by using extrusion or subdivision and then modify them. For information about how to add a primitive object to your scene, see [Create and import 3D objects](#). For information about how to add more vertices to an object, see [Modify objects](#).

Work with the Model Editor

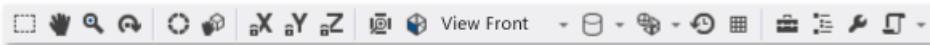
The following sections describe how to use the Model Editor to work with 3D models.

Model Editor toolbars

The Model Editor toolbars contain commands that help you work with 3D models.

Commands that affect the state of the Model Editor are located on the **Model Editor Mode** toolbar in the main Visual Studio window. Modeling tools and scripted commands are located on the **Model Editor** toolbar on the Model Editor design surface.

Here's the **Model Editor Mode** toolbar:



This table describes the items on the **Model Editor Mode** toolbar, which are listed in the order in which they appear from left to right.

TOOLBAR ITEM	DESCRIPTION
Select	Enables selection of points, edges, faces, or objects in the scene, depending on the active selection mode.
Pan	Enables movement of a 3D scene relative to the window frame. To pan, select a point in the scene and move it around. In Select mode, you can press and hold Ctrl to activate Pan mode temporarily.
Zoom	Enables the display of more or less scene detail relative to the window frame. In Zoom mode, select a point in the scene and then move it right or down to zoom in, or left or up to zoom out. In Select mode, you can zoom in or out by using the mouse wheel while you press and hold Ctrl .
Orbit	Positions the view on a circular path around the selected object. If no object is selected, the path is centered on the scene origin. Note: This mode has no effect when Orthographic projection is enabled.
World Local	When this item is enabled, transformations on the selected object occur in world-space. Otherwise, transformations on the selected object occur in local-space.

TOOLBAR ITEM	DESCRIPTION
Pivot Mode	When this item is enabled, transformations affect the location and orientation of the <i>pivot point</i> of the selected object (The pivot point defines the center of translation, scaling, and rotation operations.) Otherwise, transformations affect the location and orientation of the object's geometry, relative to the pivot point.
Lock X axis	Restricts object manipulation to the x axis. Applies only when you use the center part of the manipulator widget.
Lock Y axis	Restricts object manipulation to the y axis. Applies only when you use the center part of the manipulator widget.
Lock Z axis	Restricts object manipulation to the z axis. Applies only when you use the center part of the manipulator widget.
Frame Object	Frames the selected object so that it's located in the center of the view.
View	<p>Sets the view orientation. Here are the available orientations:</p> <p>Front Positions the view in front of the scene.</p> <p>Back Positions the view behind the scene.</p> <p>Left Positions the view to the left of the scene.</p> <p>Right Positions the view to the right of the scene.</p> <p>Top Positions the view above the scene.</p> <p>Bottom Positions the view beneath the scene. Note: This is the only way to change the view direction when Orthographic projection is enabled.</p>
Projection	<p>Sets the kind of projection that is used to draw the scene. Here are the available projections:</p> <p>Perspective In perspective projection, objects that are farther away from the viewpoint appear smaller in size and ultimately converge to a point in the distance.</p> <p>Orthographic In Orthographic projection, objects appear to be the same size, regardless of their distance from the viewpoint. No convergence is displayed. When Orthographic projection enabled, you can't use Orbit mode to position the view.</p>

TOOLBAR ITEM	DESCRIPTION
Draw Style	<p>Sets how objects in the scene are rendered. Here are the available styles:</p> <p>Wire Frame When enabled, objects are rendered as wireframes.</p> <p>Overdraw When enabled, objects are rendered by using additive blending. You can use this to visualize how much overdraw is occurring in the scene.</p> <p>Flat Shaded When enabled, objects are rendered by using a basic, flat shaded lighting model. You can use this to see the faces of an object more easily.</p> <p>If none of these options are enabled, each object is rendered by using the material that's applied to it.</p>
Real-Time Rendering Mode	<p>When real-time rendering is enabled, Visual Studio redraws the design surface, even when no user action is performed. This mode is useful when you work with shaders that change over time.</p>
Toggle Grid	<p>When this item is enabled, a grid is displayed. Otherwise, the grid is not displayed.</p>
Toolbox	<p>Alternately shows or hides the Toolbox.</p>
Document Outline	<p>Alternately shows or hides the Document Outline window.</p>
Properties	<p>Alternately shows or hides the Properties window.</p>
Advanced	<p>Contains advanced commands and options.</p> <p>Graphics Engines</p> <p>Render with D3D11 Uses Direct3D 11 to render the Model Editor design surface.</p> <p>Render with D3D11WARP Uses Direct3D 11 Windows Advanced Rasterization Platform (WARP) to render the Model Editor design surface.</p> <p>Scene Management</p> <p>Import Imports objects from another 3D model file to the current scene.</p> <p>Attach to Parent Establishes the first of multiple selected objects as the parent of the remaining selected objects.</p> <p>Detach from Parent Detaches the selected object from its parent. The selected object becomes a <i>root object</i> in the scene. A root object doesn't have a parent object.</p>

TOOLBAR ITEM	<p>Create Group DESCRIPTION Groups the selected objects as sibling objects.</p> <p>Merge Objects Combines the selected objects into one object.</p> <p>Create New Object From Polygon Selection Removes the selected faces from the current object and adds to the scene a new object that contains those faces.</p> <p>Tools</p> <p>Flip Polygon Winding Flips the selected polygons so that its winding order and surface normal are inverted.</p> <p>Remove All Animation Removes animation data from the objects.</p> <p>Triangulate Converts the selected object to triangles.</p> <p>View</p> <p>Backface Culling Enables or disables backface culling.</p> <p>Frame Rate Displays the frame rate in the upper-right corner of the design surface. The frame rate is the number of frames that are drawn per second.</p> <p>This option is useful when you enable the Real-Time Rendering Mode option.</p> <p>Show All Shows all objects in the scene. This resets the Hidden property of each object to False.</p> <p>Show Face Normals Shows the normal of each face.</p> <p>Show Missing Materials Displays a special texture on objects that don't have a material assigned to them.</p> <p>Show Pivot Enables or disables the display of a 3D axis marker at the pivot point of the active selection.</p> <p>Show Placeholder Nodes Shows placeholder nodes. A placeholder node is created when you group objects.</p> <p>Show Vertex Normals Shows the normal of each vertex. Tip: You can choose the Scripts button to run the last script again.</p>
---------------------	--

Here's the **Model Editor** toolbar:



The next table describes the items on the **Model Editor** toolbar, which are listed in the order in which they appear from top to bottom.

TOOLBAR ITEM	DESCRIPTION
Translate	Moves the selection.
Scale	Changes the size of the selection.
Rotate	Rotates the selection.
Select Point	Sets the Selection mode to select individual points on an object.
Select Edge	Sets the Selection mode to select an edge (a line between two vertices) on an object.
Select Face	Sets the Selection mode to select a face on an object.
Select Object	Sets the Selection mode to select an entire object.
Extrude	Creates an additional face and connects it to the selected face.
Subdivide	Divides each selected face into multiple faces. To create the new faces, new vertices are added—one in the center of the original face, and one in the middle of each edge—and then joined together with the original vertices. The number of added faces is equal to the number of edges in the original face.

Control the view

The 3D scene is rendered according to the view, which can be thought of as a virtual camera that has a position and an orientation. To change the position and orientation, use the view controls on the **Model Editor Mode** toolbar.

The following table describes the primary view controls.

VIEW CONTROL	DESCRIPTION
Pan	Enables movement of a 3D scene relative to the window frame. To pan, select a point in the scene and move it around. In Select mode, you can press and hold Ctrl to activate Pan mode temporarily.

VIEW CONTROL	DESCRIPTION
Zoom	Enables the display of more or less scene detail relative to the window frame. In Zoom mode, select a point in the scene and then move it right or down to zoom in, or left or up to zoom out. In Select mode, you can zoom in or out by using the mouse wheel while you press and hold Ctrl .
Orbit	Positions the view on a circular path around the selected object. If no object is selected, the path is centered on the scene origin. Note: This mode has no effect when Orthographic projection is enabled.
Frame Object	Frames the selected object so that it's located in the center of the view.

The view is established by the virtual camera, but it's also defined by a projection. The projection defines how shapes and objects in the view are translated into pixels on the design surface. On the **Model Editor** toolbar, you can choose either **Perspective** or **Orthographic** projection.

PROJECTION	DESCRIPTION
Perspective	In perspective projection, objects that are farther away from the viewpoint appear smaller in size and ultimately converge to a point in the distance.
Orthographic	In Orthographic projection, objects appear to be the same size, regardless of their distance from the viewpoint. No convergence is displayed. When Orthographic projection enabled, you can't use Orbit mode to position the view arbitrarily.

You might find it useful to view a 3D scene from a known position and angle, for example, when you want to compare two similar scenes. For this scenario, the Model Editor provides several predefined views. To use a predefined view, on the **Model Editor Mode** toolbar, choose **View**, and then choose the predefined view you want—front, back, left, right, top, or bottom. In these views, the virtual camera looks directly at the origin of the scene. For example, if you choose **View Top**, the virtual camera looks at the origin of the scene from directly above it.

View additional geometry details

To better understand a 3D object or scene, you can view additional geometry details such as per-vertex normals, per-face normals, the pivot points of the active selection, and other details. To enable or disable them, on the **Model Editor** toolbar, choose **Scripts > View**, and then choose the one you want.

Create and import 3D objects

To add a predefined 3D shape to the scene, in the **Toolbox**, select the one you want and then move it to the design surface. New shapes are positioned at the origin of the scene. The Model Editor provides seven shapes: **Cone**, **Cube**, **Cylinder**, **Disc**, **Plane**, **Sphere**, and **Teapot**.

To import a 3D object from a file, on the **Model Editor** toolbar, choose **Advanced > Scene Management > Import** > and then specify the file that you want to import.

Transform objects

You can *transform* an object by changing its **Rotation**, **Scale**, and **Translation** properties. *Rotation* orients an

object by applying successive rotations around the x-axis, y-axis, and z-axis defined by its pivot point. Each rotation specification has three components—x, y, and z, in that order—and the components are specified in degrees. **Scaling** resizes an object by stretching it by a specified factor along one or more axes centered on its pivot point. **Translation** locates an object in 3-dimensional space relative to its parent instead of its pivot point.

You can transform an object either by using modeling tools or by setting properties.

Transform an object by using modeling tools

1. In **Select** mode, select the object you want to transform. A wireframe overlay indicates that the object is selected.
2. On the **Model Editor** toolbar, choose the **Translate**, **Scale**, or **Rotate** tool. A translation, scaling, or rotation manipulator appears for the selected object.
3. Use the manipulator to perform the transformation. For translation and scaling transformations, the manipulator is an axis indicator. You can change one axis at a time, or you can change all axes at the same time by using the white cube at the center of the indicator. For rotation, the manipulator is a sphere made of color-coded circles that correspond to the x-axis (red), y-axis (green), and z-axis (blue). You have to change each axis individually to create the rotation you want.

Transform an object by setting its properties

1. In **Select** mode, select the object that you want to transform. A wireframe overlay indicates that the object is selected.
2. In the **Properties** window, specify values for the **Rotation**, **Scale**, and **Translation** properties.

IMPORTANT

For the **Rotation** property, specify the degree of rotation around each of the three axes. Rotations are applied in order, so make sure to plan a rotation, first in terms of the x-axis rotation, then the y-axis, and then the z-axis.

By using the modeling tools, you can create transformations quickly but not precisely. By setting the object properties, you can specify transformations precisely but not quickly. We recommend that you use the modeling tools to get "close enough" to the transformations you want, and then fine-tune the property values.

If you don't want to use manipulators, you can enable free-form mode. On the **Model Editor** toolbar, choose **Scripts > Tools > Free-form Manipulation** to enable (or disable) free-form mode. In free-form mode, you can begin a manipulation at any point on the design surface instead of a point on the manipulator. In free-form mode, you can constrain changes to certain axes by locking the ones you don't want to change. On the **Model Editor Mode** toolbar, choose any combination of the **Lock X**, **Lock Y**, and **Lock Z** buttons.

You might find it useful to work with objects by using snap-to-grid. On the **Model Editor Mode** toolbar, choose **Snap** to enable (or disable) snap-to-grid. When snap-to-grid is enabled, translation, rotation, and scaling transformations are constrained to predefined increments.

Work with the pivot point

The pivot point of an object defines its center of rotation and scaling. You can change the pivot point of an object to change how it's affected by rotation and scaling transformations. On the **Model Editor Mode** toolbar, choose **Pivot Mode** to enable (or disable) pivot mode. When pivot mode is enabled, a small axis indicator appears at the pivot point of the selected object. You can then use the **Translation** and **Rotation** tools to manipulate the pivot point.

For a demonstration that shows how to use the pivot point, see [How to: Modify the pivot point of a 3D model](#).

World and local modes

Translation and rotation can occur in either the local coordinate system (or *local frame-of-reference*) of the object, or in the coordinate system of the world (or the *world frame-of-reference*). The world frame-of-reference is

independent of the rotation of the object. Local mode is the default. To enable (or disable) world mode, on the **Model Editor Mode** toolbar, choose the **WorldLocal** button.

Modify objects

You can change the shape of a 3D object by moving or deleting its vertices, edges, and faces. By default, the Model Editor is in *object mode*, so that you can select and transform entire objects. To select points, edges, or faces, choose the appropriate selection mode. On the **Model Editor Mode** toolbar, choose **Selection modes**, and then choose the mode that you want.

You can create additional vertices by extrusion or by subdivision. Extrusion duplicates the vertices of a face (a co-planar set of vertices), which remain connected by the duplicated vertices. Subdivision adds vertices to create several faces where there was previously one. To create the new faces, new vertices are added—one in the center of the original face, and one in the middle of each edge—and then joined together with the original vertices. The number of added faces is equal to the number of edges in the original face. In both cases, you can translate, rotate, and scale the new vertices to change the geometry of the object.

To extrude a face from an object

1. In face-select mode, select the face you want to extrude.
2. On the **Model Editor** toolbar, choose **Scripts > Tools > Extrude**.

To subdivide faces

1. In face-select mode, select the faces you want to subdivide. Because subdivision creates new edge data, subdividing all faces at once gives more-consistent results when the faces are adjacent.
2. On the **Model Editor** toolbar, choose **Scripts > Tools > Subdivide**.

You can also triangulate faces, merge objects, and convert polygon selections into new objects. Triangulation creates additional edges such that a non-triangular face is converted to an optimal number of triangles; however, it doesn't provide additional geometric detail. Merging combines selected objects into one object. New objects can be created from a polygon selection.

Triangulate a face

1. In face-select mode, select the face you want to triangulate.
2. On the **Model Editor** toolbar, choose **Scripts > Tools > Triangulate**.

Merge objects

1. In object-select mode, select the objects you want to merge.
2. On the **Model Editor** toolbar, choose **Scripts > Tools > Merge Objects**.

Create an object from a polygon selection

1. In face-select mode, select the faces you want to create a new object from.
2. On the **Model Editor** toolbar, choose **Scripts > Tools > Create New Object from Polygon Selection**.

Work with materials and shaders

The appearance of an object is determined by the interaction of lighting in the scene and the material of the object. Materials are defined by properties that describe how the surface reacts to different types of light and by a shader program that calculates the final color of each pixel on the object surface based on lighting information, texture maps, normal maps, and other data.

The Model Editor provides these default materials:

MATERIAL	DESCRIPTION
Unlit	Renders a surface without any simulated lighting.

MATERIAL	DESCRIPTION
Lambert	Renders a surface with simulated ambient lighting and diffuse lighting.
Phong	Renders a surface with simulated ambient lighting, diffuse lighting, and specular highlights.

Each of these materials applies one texture on the surface of an object. You can set a different texture for each object that uses the material.

To modify how a particular object reacts to the different light sources in the scene, you can change the lighting properties of material independent of other objects that use the material. This table describes common lighting properties:

LIGHTING PROPERTY	DESCRIPTION
Ambient	Describes how the surface is affected by ambient lighting.
Diffuse	Describes how the surface is affected by directional and point lights.
Emissive	Describes how the surface emits light, independent of other lighting.
Specular	Describes how the surface reflects directional and point lights.
Specular Power	Describes the breadth and intensity of specular highlights.

Depending on what a material supports, you can change its lighting properties, textures, and other data. In **Select** mode, select the object whose material you want to change, and then in the **Properties** window, change the **MaterialAmbient**, **MaterialDiffuse**, **MaterialEmissive**, **MaterialSpecular**, **MaterialSpecularPower**, or other available property. A material can expose up to eight textures, whose properties are named sequentially from **Texture1** to **Texture8**.

To remove all materials from an object, on the **Model Editor** toolbar, choose **Scripts > Materials > Remove Materials**.

You can use the **Shader Designer** to create custom shader materials that you can apply to objects in your 3D scene. For information about how to create custom shader materials, see [Shader Designer](#). For information about how to apply a custom shader material to an object, see [How to: Apply a shader to a 3D model](#).

Scene management

You can manage scenes as a hierarchy of objects. When multiple objects are arranged in a hierarchy, any translation, scale, or rotation of a parent node also affects its children. This is useful when you want to construct complex objects or scenes from more basic objects.

You can use the **Document Outline** window to view the scene hierarchy and select scene nodes. When you select a node in the outline, you can use the **Properties** window to modify its properties.

You can construct a hierarchy of objects either by making one of them the parent to the others or by grouping them together as siblings under a placeholder node that acts as the parent.

Create a hierarchy that has a parent object

1. In **Select** mode, select two or more objects. The first one you select will be the parent object.

2. On the **Model Editor** toolbar, choose **Scripts > Scene Management > Attach to Parent**.

Create a hierarchy of sibling objects

1. In **Select** mode, select two or more objects. A placeholder object is created and becomes their parent object.

2. On the **Model Editor** toolbar, choose **Scripts > Scene Management > Create Group**.

The Model Editor uses a white wireframe to identify the first selected object, which becomes the parent. Other objects in the selection have a blue wireframe. By default, placeholder nodes are not displayed. To display placeholder nodes, on the **Model Editor** toolbar, choose **Scripts > Scene Management > Show Placeholder Nodes**. You can work with placeholder nodes just as you work with non-placeholder objects.

To remove the parent-child association between two objects, select the child object, and then on the **Model Editor** toolbar, choose **Scripts > Scene Management > Detach from Parent**. When you detach the parent from a child object, the child object becomes a root object in the scene.

Keyboard shortcuts

COMMAND	KEYBOARD SHORTCUTS
Switch to Select mode	Ctrl+G, Ctrl+Q S
Switch to Zoom mode	Ctrl+G, Ctrl+Z Z
Switch to Pan mode	Ctrl+G, Ctrl+P K
Select all	Ctrl+A
Delete the current selection	Delete
Cancel the current selection	Escape (Esc)
Zoom in	Mouse wheel forward Ctrl+Mouse wheel forward Shift+Mouse wheel forward Ctrl+PageUp Plus Sign (+)
Zoom out	Mouse wheel backward Ctrl+Mouse wheel backward Shift+Mouse wheel backward Ctrl+PageDown Minus Sign (-)

COMMAND	KEYBOARD SHORTCUTS
Pan the camera up	PageDown
Pan the camera down	PageUp
Pan the camera left	Mouse wheel left Ctrl+PageDown
Pan the camera right	Mouse wheel right Ctrl+PageDown
View top of model	Ctrl+L, Ctrl+T T
View bottom of model	Ctrl+L, Ctrl+U
View left side of model	Ctrl+L, Ctrl+L
View right side of model	Ctrl+L, Ctrl+R
View front of model	Ctrl+L, Ctrl+F
View back of model	Ctrl+L, Ctrl+B
Frame object in window	F
Toggle wireframe mode	Ctrl+L, Ctrl+W
Toggle snap-to-grid	Ctrl+G, Ctrl+N
Toggle pivot mode	Ctrl+G, Ctrl+V
Toggle x-axis restriction	Ctrl+L, Ctrl+X
Toggle y-axis restriction	Ctrl+L, Ctrl+Y
Toggle z-axis restriction	Ctrl+L, Ctrl+Z
Switch to translation mode	Ctrl+G, Ctrl+W W
Switch to scale mode	Ctrl+G, Ctrl+E E
Switch to rotation mode	Ctrl+G, Ctrl+R R

COMMAND	KEYBOARD SHORTCUTS
Switch to point-select mode	Ctrl+L, Ctrl+1
Switch to edge-select mode	Ctrl+L, Ctrl+2
Switch to face-select mode	Ctrl+L, Ctrl+3
Switch to object-select mode	Ctrl+L, Ctrl+4
Switch to orbit (camera) mode	Ctrl+G, Ctrl+O
Select next object in scene	Tab
Select previous object in scene	Shift+Tab
Manipulate the selected object based on the current tool.	The Arrow keys
Deactivate current manipulator	Q
Rotate camera	Alt+Drag with left mouse button

Related topics

TITLE	DESCRIPTION
Working with 3D assets for games and apps	Provides an overview of the Visual Studio tools that you can use to work with graphics assets such as textures and images, 3D models, and shader effects.
Image Editor	Describes how to use the Visual Studio Image Editor to work with textures and images.
Shader Designer	Describes how to use the Visual Studio Shader Designer to work with shaders.

How to: Create a basic 3D model

10/18/2019 • 2 minutes to read • [Edit Online](#)

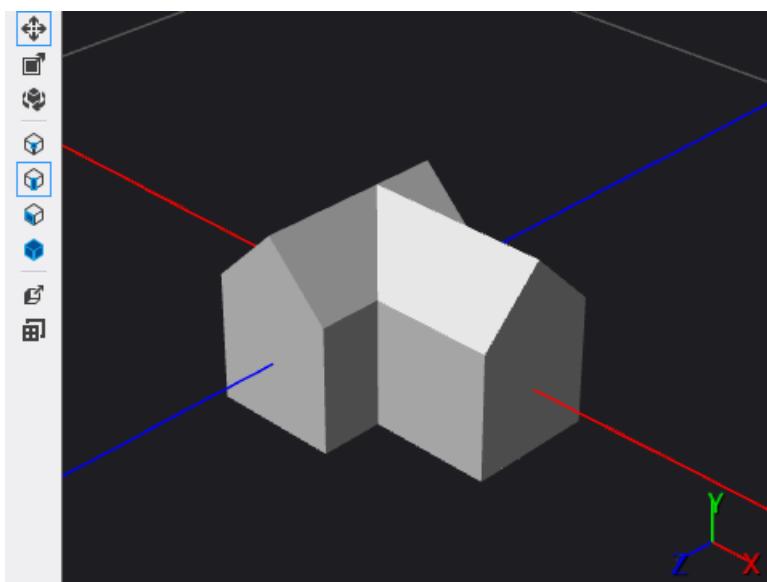
This article demonstrates how to use the Model Editor to create a basic 3D model. The following activities are covered:

- Adding objects to a scene
- Selecting faces and edges
- Translating selections
- Using the **Subdivide face** and **Extrude face** tools
- Using the **Triangulate** command

Create a basic 3D model

You can use the Model Editor to create and modify 3D models and scenes for your game or app. The following steps show how to use the Model Editor to create a simplified 3D model of a house. A simplified model can be used as a stand-in for final art assets that are still being created, as a mesh for collision detection, or as a low-detail model to be used when the object that it represents is too far away to benefit from more detailed rendering.

When you're finished, the model should look like this:

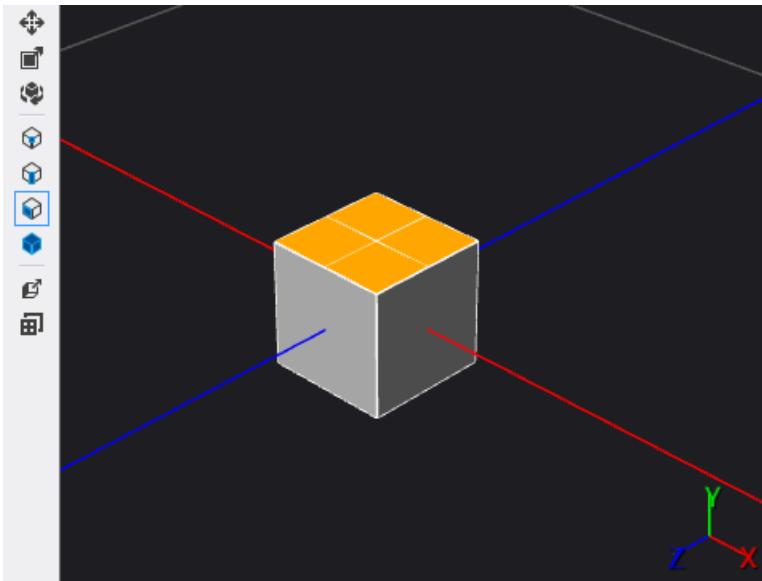


Before you begin, make sure that the **Properties** window and **Toolbox** are displayed.

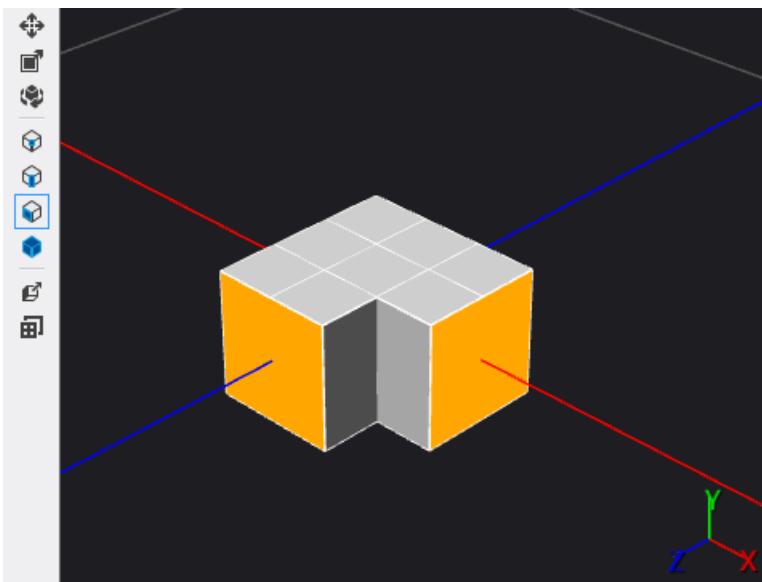
To create a simplified 3D model of a house

1. Create a 3D model with which to work. For information about how to add a model to your project, see the Getting Started section in [Model Editor](#).
2. Add a cube to the scene. In the **Toolbox** window, under **Shapes**, select **Cube** and then move it to the design surface.
3. Switch to face-selection. On the Model Editor toolbar, choose **Select Face**.
4. Subdivide the top of the cube. In face selection mode, choose the cube once to activate it for selection, and then choose the top of the cube to select the top face. On the Model Editor toolbar, choose **Subdivide face**.

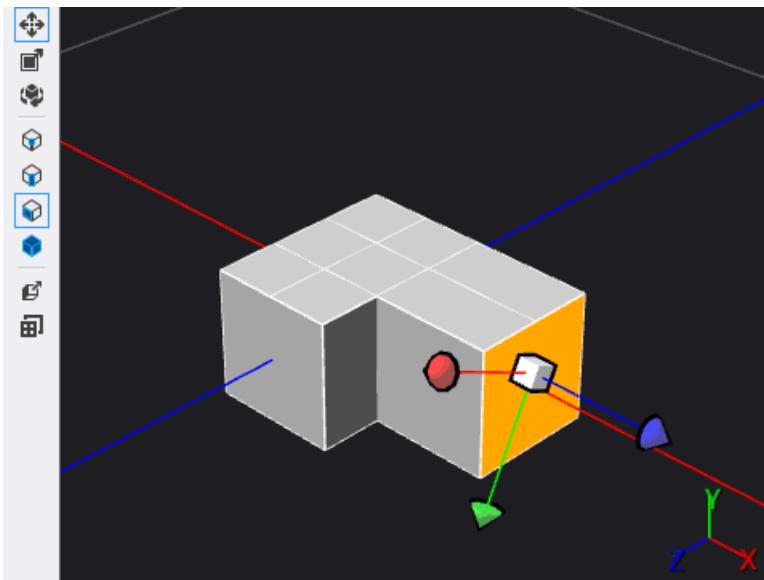
This adds new vertices to the top of the cube that split it into four equally sized partitions.



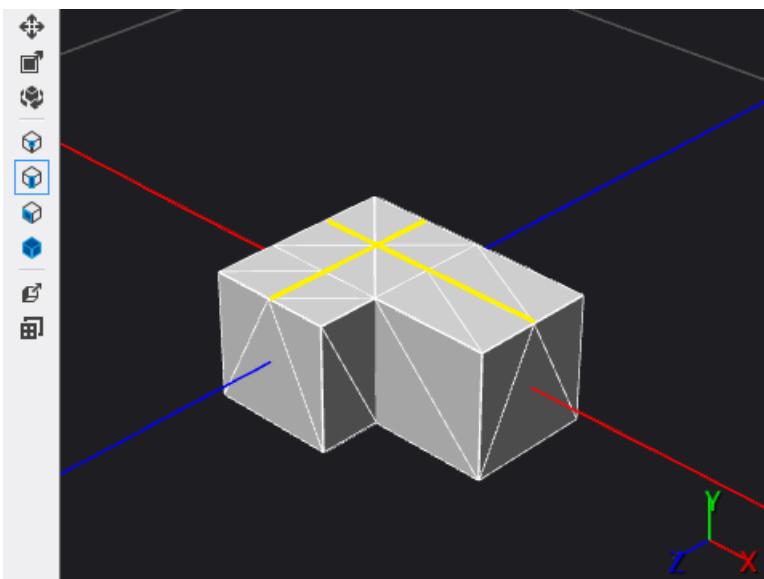
5. Extrude two adjacent sides of the cube—for example, the front and right sides of the cube. In face selection mode, choose the cube once to activate it for selection and then choose one side of the cube. Press and hold the **Ctrl** key, choose another side of the cube that is adjacent to the side you selected first, and then on the Model Editor toolbar, choose **Extrude face**.



6. Extend one of the extrusions. Choose one of the faces that you just extruded, and then, on the Model Editor toolbar, choose the **Translate** tool and move the translation manipulator in the same direction as the extrusion.

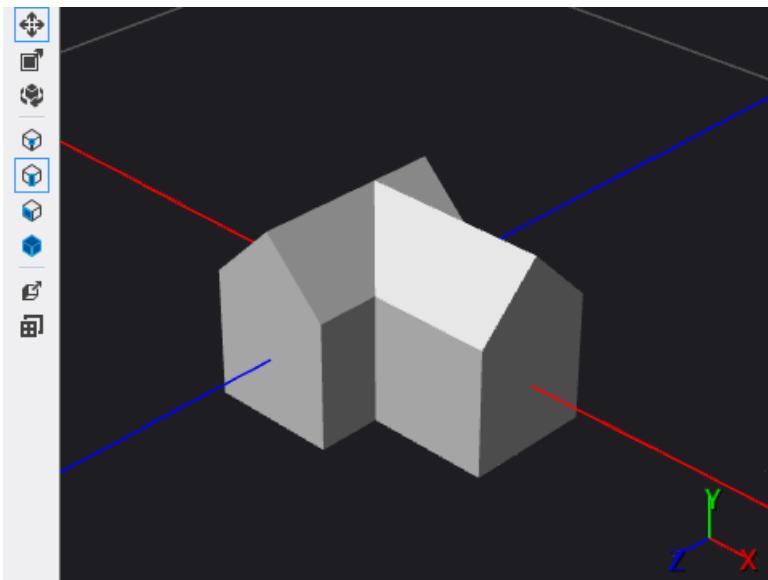


7. Triangulate the model. On the Model Editor toolbar, choose **Advanced > Tools > Triangulate**.
8. Create the roof of the house. Switch to edge-selection mode by choosing **Select Edge** on the Model Editor toolbar, and then choose the cube to activate it. Press and hold the **Ctrl** key as you select the edges that are shown here:



When the edges are selected, on the Model Editor toolbar, choose the **Translate** tool and then move the translation manipulator upward to create the roof of the house.

The simplified house model is complete. Here's the final model again, with flat shading applied:



As a next step, you can apply a shader to this 3D model. For information, see [How to: Apply a shader to a 3D model](#).

See also

- [How to: Model 3D terrain](#)
- [Model editor](#)
- [Shader designer](#)

How to: Modify the pivot point of a 3D model

10/18/2019 • 2 minutes to read • [Edit Online](#)

This article demonstrates how to use the Model Editor to modify the *pivot point* of a 3D model. The pivot point is the point in space that defines the mathematical center of the object for rotation and scaling.

Modify the pivot point of a 3D model

You can redefine the origin of a 3D model by modifying its pivot point.

Make sure that the **Properties** window and the **Toolbox** are displayed.

1. Begin with an existing 3D model, such as the one that's described in [How to: Create a basic 3D model](#).
2. Enter pivot mode. On the **Model Editor Mode** toolbar, choose the **Pivot Mode** button to activate pivot mode. A box appears around the **Pivot Mode** button to indicate that the Model Editor is now in pivot mode. In pivot mode, operations such as translation affect the pivot point of the object instead of the structure of the object in world-space.
3. Modify the pivot point of the object. In **Select** mode, select the object, and then on the **Model Viewer** toolbar, choose the **Translate** tool. A box that represents the pivot point appears on the design surface. Move the box to modify the pivot point of the object.

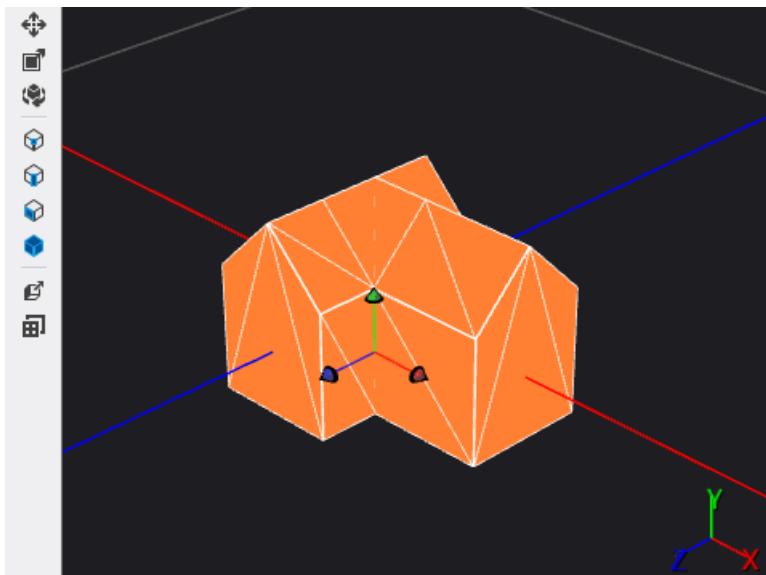
By moving the box, you can move the pivot point in all three dimensions. To translate the pivot point along one axis, move the arrow that corresponds to that axis. The box and arrows change to a yellow color to indicate the axis that's affected by the translation.

You can also specify the pivot point by using the **Pivot Translation** property in the **Properties** window.

TIP

You can view the effect of the new pivot point by rotating the object. To rotate it, use the **Rotate** tool or modify the **Rotation** property.

Here's a model that has a modified pivot point:



See also

- [How to: Create a basic 3D model](#)
- [Model editor](#)

How to: Model 3D terrain

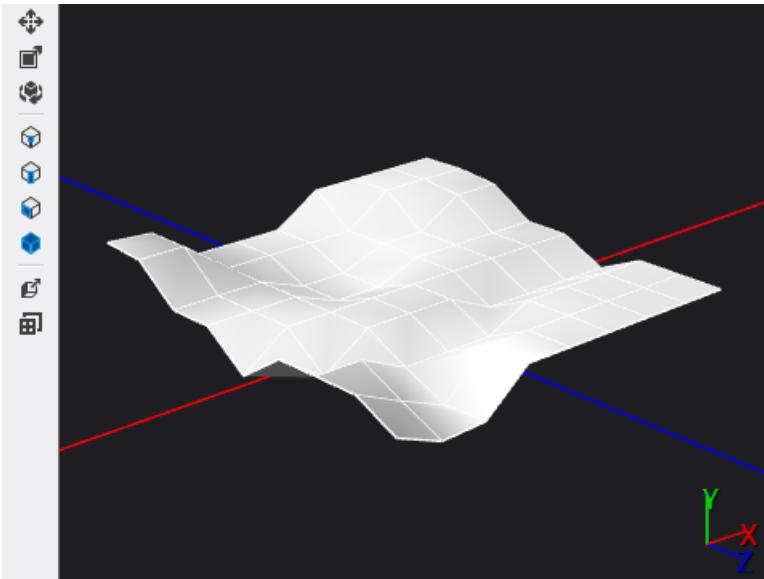
10/18/2019 • 2 minutes to read • [Edit Online](#)

This article demonstrates how to use the Model Editor to create a 3D terrain model.

Create a 3D terrain model

You can create a 3D terrain by subdividing a plane to make additional faces, and then manipulating their vertices to create interesting terrain features.

When you're finished, the model should look like this:



Before you begin, make sure that the **Properties** window and **Toolbox** are displayed.

1. Create a 3D model with which to work. For information about how to add a model to your project, see the Getting Started section in [Model editor](#).
2. Add a plane to the scene. In the **Toolbox**, under **Shapes**, select **Plane** and move it to the design surface.

TIP

To make the plane object easier to work with, you can frame it in the design surface. In **Select** mode, select the plane object, and then on the Model Editor toolbar, choose the **Frame Object** button.

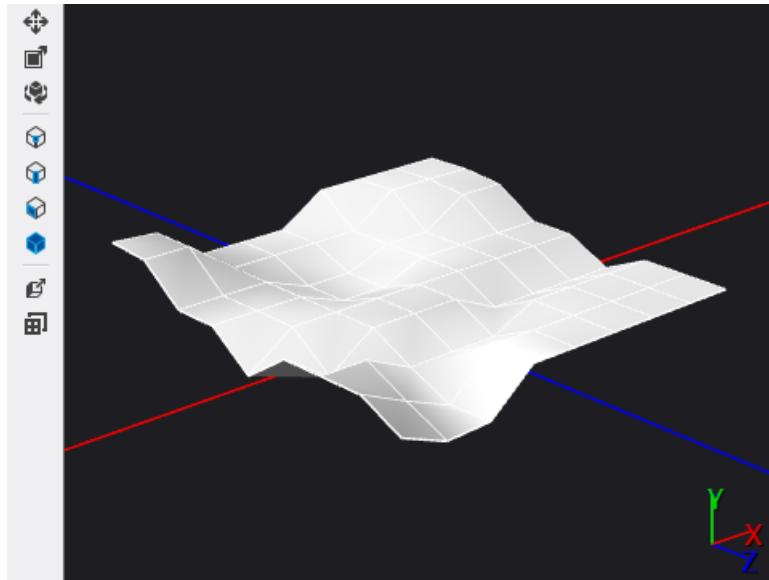
3. Enter face selection mode. On the Model Editor toolbar, choose **Select Face**.
4. Subdivide the plane. In face selection mode, choose the plane once to activate it for selection, and then choose it again to select its only face. On the Model Editor toolbar, choose **Subdivide face**. This adds new vertices to the plane that split it into four equally sized partitions.
5. Create more subdivisions. With the new faces still selected, choose **Subdivide face** two more times. This creates a total of 64 faces. By creating more subdivisions, you can give the terrain even more detail.
6. Enter point selection mode. On the Model Editor toolbar, choose **Select Point**.
7. Modify a point to create a terrain feature. In point selection mode, select one of the points, and then on the Model Editor toolbar, choose the **Translate** tool. A box that represents the point appears on the design

surface. Use the green arrow to move the box and thereby modify the height of the point. Repeat this step for different points to create interesting terrain features.

TIP

You can select several points at once to modify them in a uniform manner.

The terrain model is complete. Here's the final model again, with Phong shading applied:



You can use this terrain model to demonstrate the effect of the gradient shader that's described in [How to: Create a geometry-based gradient shader](#).

See also

- [Model editor](#)

How to: Apply a shader to a 3D model

10/18/2019 • 2 minutes to read • [Edit Online](#)

This article demonstrates how to use the Model Editor to apply a Directed Graph Shader Language (DGSL) shader to a 3D model.

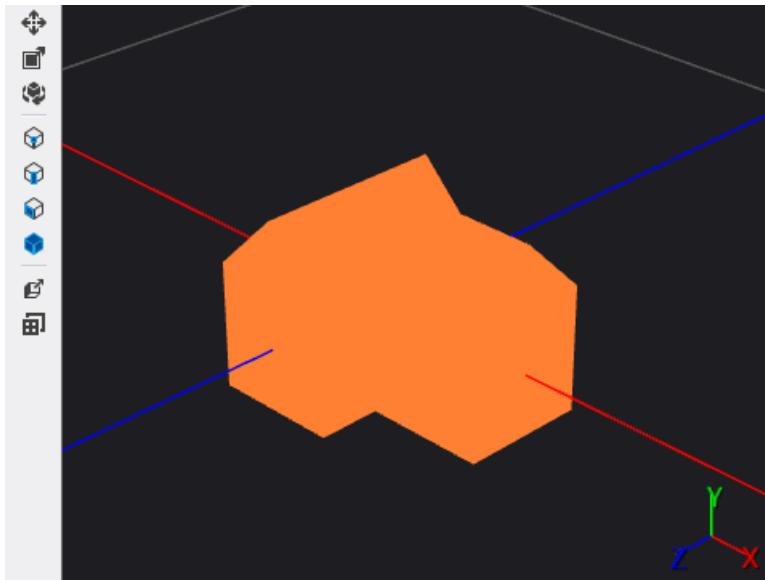
Apply a shader to a 3D model

You can apply a shader effect to a 3D model to give it an interesting appearance.

Before you begin, make sure that the **Properties** window is displayed.

1. Begin with a 3D scene that contains one or more models. If you don't have a suitable 3D scene, create one as described in [How to: Create a basic 3D Model](#). You must also have a DGSL shader that you can apply to the model. If you don't have a suitable shader, create one as described in [How to: Create a basic color shader](#) and make sure that you've saved it to a file before you continue.
2. In **Select** mode, select the model to which you want to apply the shader, and then in the **Properties** window, in the **Filename** property of the **Effect** property group, specify the DGSL shader that you want to apply to the model.

Here's a model that has the basic color effect applied to it:



After you apply a shader to a model, you can open it in the Shader Designer by selecting the model, and then in the **Properties** window, in the **(Advanced)** property of the **Effect** property group, choosing the ellipsis (...) button.

See also

- [How to: Create a basic 3D model](#)
- [How to: Create a basic color shader](#)
- [Model editor](#)
- [Shader designer](#)

Work with shaders

10/18/2019 • 2 minutes to read • [Edit Online](#)

You can use the graph-based Shader Designer in Visual Studio to design custom shader effects. You can use these shaders in your DirectX-based game or app.

Shaders

A *shader* is a computer program that performs graphics calculations—for example, vertex transformations or pixel coloring—and typically runs on a graphics processing unit (GPU) instead of the CPU. Because most stages of the traditional, fixed-function graphics pipeline are now performed by shader programs, you can use them to create a pipeline that's specific to the needs of your app.

The most common kinds of shaders are *vertex shaders*, which perform per-vertex calculations and replace the fixed-function transformation and lighting circuitry in non-programmable graphics hardware, and *pixel shaders*, which perform per-pixel calculations that determine the color of a pixel and replace the fixed-function color-combiner circuitry in non-programmable graphics hardware. Modern graphics hardware has made even more kinds of shaders possible—*hull shaders*, *domain shaders*, and *geometry shaders* for graphics calculations, and *compute shaders* for non-graphics computations. None of these stages are even available in non-programmable graphics hardware. Shaders were originally created by using an assembly-like language that provided data-parallel (SIMD) and graphics-centric (dot product) instructions. Now, shaders are typically created by using high-level, C-like languages like HLSL (High Level Shader Language).

You can use the Shader Designer to create pixel shaders interactively instead of by entering and compiling code. In the Shader Designer, a shader is defined by a number of nodes that represent data and operations, and connections between nodes that represent the flow of data values and intermediate results through the shader. By using this approach and the real-time preview in the Shader Designer, you can visualize the execution of the shader more easily, and "discover" interesting shader variations through experimentation.

DGSL documents

The Shader Designer saves shaders in the Directed Graph Shader Language (DGSL) format, which is an XML format that's based on Directed Graph Markup Language (DGML). You can apply DGSL shaders directly to 3D models in the Model Editor. However, before you can use a DGSL shader in your app, you must export it to a format that DirectX understands—for example, HLSL.

Because DGSL is compatible with DGML, you can use tools that are designed to analyze DGML documents to analyze your DGSL shaders. For information about DGML, see [Understanding Directed Graph Markup Language \(DGML\)](#).

Related topics

TITLE	DESCRIPTION
Shader Designer	Describes how to use the Visual Studio Shader Designer to work with shaders.
Shader Designer nodes	Discusses the kinds of Shader Designer nodes that you can use to achieve graphics effects.

TITLE	DESCRIPTION
Shader Designer examples	Provides links to topics that demonstrate how to use the Shader Designer to achieve common graphics effects.

Shader Designer

10/18/2019 • 10 minutes to read • [Edit Online](#)

This document describes how to work with the Visual Studio **Shader Designer** to create, modify, and export custom visual effects that are known as *shaders*.

You can use **Shader Designer** to create custom visual effects for your game or app even if you don't know high-level shader language (HLSL) programming. To create a shader in **Shader Designer**, you lay it out as a graph. That is, you add to the design surface *nodes* that represent data and operations and then make connections between them to define how the operations process the data. At each operation node, a preview of the effect up to that point is provided so that you can visualize its result. Data flows through the nodes toward a final node that represents the output of the shader.

Supported formats

The **Shader Designer** supports these shader formats:

FORMAT NAME	FILE EXTENSION	SUPPORTED OPERATIONS (VIEW, EDIT, EXPORT)
Directed Graph Shader Language	.dgsL	View, Edit
HLSL Shader (source code)	.hlsl	Export
HLSL Shader (bytecode)	.cso	Export
C++ header (HLSL bytecode array)	.h	Export

Get started

This section describes how to add a DGSL shader to your Visual Studio C++ project and provides basic information to help you get started.

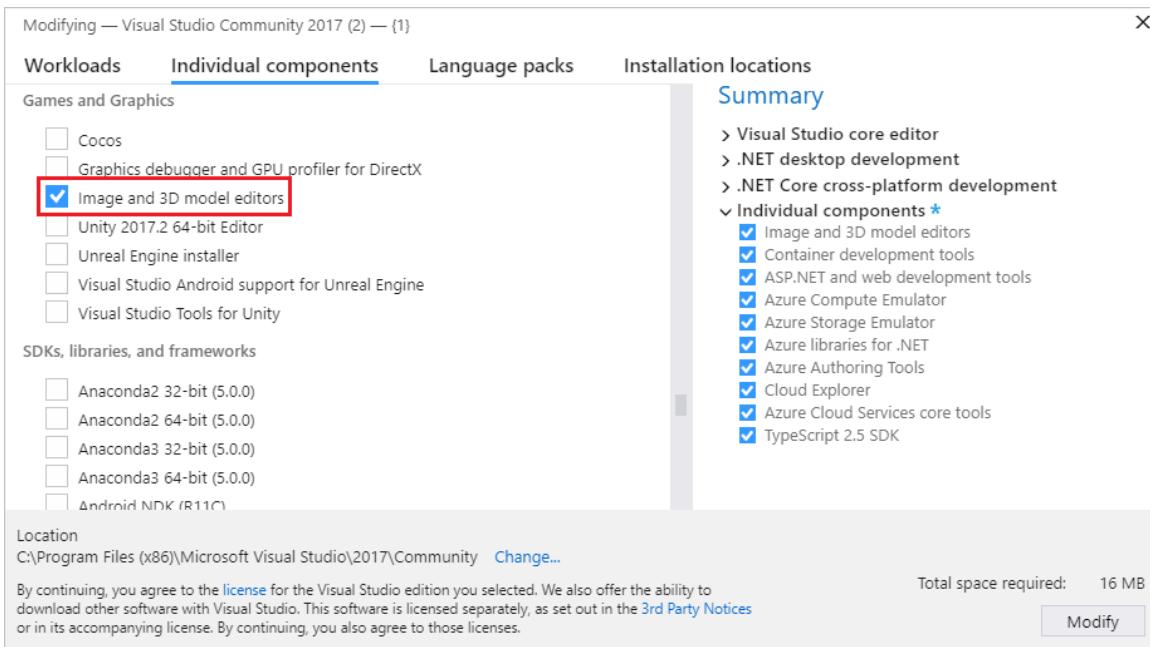
NOTE

Automatic build integration of graphics items like shader graphs (.dgsL files) is only supported for C++ projects.

To add a DGSL shader to your project

1. Ensure you have the required Visual Studio component installed that you need to work with graphics. The component is called **Image and 3D model editors**.

To install it, open Visual Studio Installer by selecting **Tools** > **Get Tools and Features** from the menu bar, and then select the **Individual components** tab. Select the **Image and 3D model editors** component under the **Games and Graphics** category, and then select **Modify**.



2. In **Solution Explorer**, open the shortcut menu for the C++ project to which you want to add the shader, and then choose **Add > New Item**.
3. In the **Add New Item** dialog box, under **Installed**, select **Graphics**, and then select **Visual Shader Graph (.dgsl)**.

NOTE

If you don't see the **Graphics** category in the **Add New Item** dialog, and you have the **Image and 3D model editors** component installed, graphics items are not supported for your project type.

4. Specify the **Name** of the shader file, and the **Location** where you want it to be created.
5. Choose the **Add** button.

The default shader

Each time that you create a DGSL shader, it begins as a minimal shader that has just a **Point Color** node that's connected to the **Final Color** node. Although this shader is complete and functional, it doesn't do much. Therefore, the first step in creating a working shader is often to delete the **Point Color** node or disconnect it from the **Final Color** node to make room for other nodes.

Work with the Shader Designer

The following sections describe how to use the Shader Designer to work with custom shaders.

Shader Designer toolbars

The Shader Designer toolbars contain commands that help you work with DGSL shader graphs.

Commands that affect the state of the Shader Designer are located on the **Shader Designer Mode** toolbar in the main Visual Studio window. Design tools and commands are located on the **Shader Designer** toolbar on the Shader Designer design surface.

Here's the **Shader Designer Mode** toolbar:



This table describes the items on the **Shader Designer Mode** toolbar, which are listed in the order in which they appear from left to right:

TOOLBAR ITEM	DESCRIPTION
Select	Enables interaction with nodes and edges in the graph. In this mode, you can select nodes and move or delete them, and you can establish edges or break them.
Pan	<p>Enables movement of a shader graph relative to the window frame. To pan, select a point on the design surface and move it around.</p> <p>In Select mode, you can press and hold Ctrl to activate Pan mode temporarily.</p>
Zoom	<p>Enables the display of more or less shader-graph detail relative to the window frame. In Zoom mode, select a point on the design surface and then move it right or down to zoom in, or left or up to zoom out.</p> <p>In Select mode, you can press and hold Ctrl to zoom in or out by using the mouse wheel.</p>
Zoom to Fit	Displays the full shader graph in the window frame.
Real-Time Rendering Mode	When real-time rendering is enabled, Visual Studio redraws the design surface, even when no user action is performed. This mode is useful when you work with shaders that change over time.
Preview with sphere	When enabled, a model of a sphere is used to preview the shader. Only one preview shape at a time can be enabled.
Preview with cube	When enabled, a model of a cube is used to preview the shader. Only one preview shape at a time can be enabled.
Preview with Cylinder	When enabled, a model of a cylinder is used to preview the shader. Only one preview shape at a time can be enabled.
Preview with cone	When enabled, a model of a cone is used to preview the shader. Only one preview shape at a time can be enabled.
Preview with teapot	When enabled, a model of a teapot is used to preview the shader. Only one preview shape at a time can be enabled.
Preview with plane	When enabled, a model of a plane is used to preview the shader. Only one preview shape at a time can be enabled.
Toolbox	Alternately shows or hides the Toolbox .
Properties	Alternatively shows or hides the Properties window.

Toolbar Item	Description
Advanced	<p>Contains advanced commands and options.</p> <p>Export: Enables the export of a shader in several formats.</p> <p>Export As: Exports the shader as either HLSL source code or as compiled shader bytecode. For more information about how to export shaders, see How to: Export a shader.</p> <p>Graphics Engines: Enables the selection of the renderer that is used to display the design surface.</p> <p>Render with D3D11: Uses Direct3D 11 to render the Shader Designer design surface.</p> <p>Render with D3D11WARP: Uses Direct3D 11 Windows Advanced Rasterization Platform (WARP) to render the Shader Designer design surface.</p> <p>View: Enables the selection of additional information about the Shader Designer.</p> <p>Frame Rate: When enabled, displays the current frame rate in the upper-right corner of the design surface. The frame rate is the number of frames that are drawn per second. This option is useful when you enable the Real-Time Rendering Mode option.</p>

TIP

You can choose the **Advanced** button to run the last command again.

Work with nodes and connections

Use **Select** mode to add, remove, reposition, connect, and configure nodes. Here's how to perform these basic operations:

To perform basic operations in Select mode

- Here's how:
 - To add a node to the graph, select it in the **Toolbox** and then move it to the design surface.
 - To remove a node from the graph, select it and then press **Delete**.
 - To reposition a node, select it and then move it to a new location.
 - To connect two nodes, move an output terminal of one node to an input terminal of the other node. Only terminals that have compatible types can be connected. A line between the terminals shows the connection.
 - To remove a connection, on the shortcut menu for either one of the connected terminals, choose **Break Links**.
 - To configure the properties of a node, select the node, and then, in the **Properties** window, specify new values for the properties.

Preview shaders

To help you understand how a shader will appear in your app, you can configure how your effect is previewed. To approximate your app, you can choose one of several shapes to render, configure textures and other

material parameters, enable animation of time-based effects, and examine the preview from different angles.

Shapes

The Shader Designer includes six shapes—a sphere, a cube, a cylinder, a cone, a teapot, and a plane—that you can use to preview your shader. Depending on the shader, certain shapes might give you a better preview.

To choose a preview shape, on the **Shader Designer Modes** toolbar, choose the shape that you want.

Textures and material parameters

Many shaders rely on textures and material properties to produce a unique appearance for each kind of object in your app. To see what your shader will look like in your app, you can set the textures and material properties that are used to render the preview to match the textures and parameters that you might use in your app.

To bind a different texture to a texture register, or to modify other material parameters:

1. In **Select** mode, select an empty area of the design surface. This causes the **Properties** window to display the global shader properties.
2. In the **Properties** window, specify new values for the texture and parameter properties that you want to change.

The following table shows the shader parameters that you can modify:

PARAMETER	PROPERTIES
Texture 1 - Texture 8	Access: Public to allow the property to be set from the Model Editor; otherwise, Private . Filename: The full path of the texture file that is associated with this texture register.
Material Ambient	Access: Public to allow the property to be set from the Model Editor; otherwise, Private . Value: The diffuse color of the current pixel due to indirect - or ambient - lighting.
Material Diffuse	Access: Public to allow the property to be set from the Model Editor; otherwise, Private . Value: A color that describes how the current pixel diffuses direct lighting.
Material Emissive	Access: Public to allow the property to be set from the Model Editor; otherwise, Private . Value: The color contribution of the current pixel due to self-provided lighting.
Material Specular	Access: Public to allow the property to be set from the Model Editor; otherwise, Private . Value: A color that describes how the current pixel reflects direct lighting.
Material Specular Power	Access: Public to allow the property to be set from the Model Editor; otherwise, Private . Value: The exponent that defines the intensity of specular highlights on the current pixel.

Time-based effects

Some shaders have a time-based component that animates the effect. To show what the effect looks like in action, the preview has to be updated several times per second. By default, the preview is only updated when the shader is changed; to change this behavior so that you can view time-based effects, you have to enable real-time rendering.

To enable real-time rendering, on the Shader Designer toolbar, choose **Real time Rendering**.

Examine the effect

Many shaders are affected by variables such as viewing angle or directional lighting. To examine how the effect responds as these variables change, you can rotate the preview shape freely and observe how the shader behaves.

To rotate the shape, press and hold **Alt**, and then select any point on the design surface and move it.

Export shaders

Before you can use a shader in your app, you have to export it in a format that DirectX understands.

You can export shaders as HLSL source code or as compiled shader bytecode. HLSL source code is exported to a text file that has an *.hsl* file name extension. Shader bytecode can be exported either to a raw binary file that has a *.cso* file name extension, or to a C++ header (*.h*) file that encodes the shader bytecode into an array.

For more information about how to export shaders, see [How to: Export a shader](#).

Keyboard shortcuts

COMMAND	KEYBOARD SHORTCUTS
Switch to Select mode	Ctrl+G , Ctrl+Q S
Switch to Zoom mode	Ctrl+G , Ctrl+Z Z
Switch to Pan mode	Ctrl+G , Ctrl+P K
Select all	Ctrl+A
Delete the current selection	Delete
Cancel the current selection	Escape (Esc)
Zoom in	Ctrl+Mouse wheel forward Plus Sign (+)
Zoom out	Ctrl+Mouse wheel backward Minus Sign (-)
Pan the design surface up	Mouse wheel backward PageDown

COMMAND	KEYBOARD SHORTCUTS
Pan the design surface down	Mouse wheel forward PageUp
Pan the design surface left	Shift+ Mouse wheel backward Mouse wheel left Shift+ PageDown
Pan the design surface right	Shift+ Mouse wheel forward Mouse wheel right Shift+ PageUp
Move the keyboard focus to another node	The Arrow keys
Select the node that has keyboard focus (adds the node to the selection group)	Shift+Spacebar
Toggle selection of the node that has keyboard focus	Ctrl+Spacebar
Toggle current selection (if no nodes are selected, select the node that has keyboard focus)	Spacebar
Move the current selection up	Shift+ Up Arrow
Move the current selection down	Shift+ Down Arrow
Move the current selection left	Shift+ Left Arrow
Move current selection right	Shift+ Right Arrow.

Related topics

TITLE	DESCRIPTION
Working with 3D assets for games and apps	Provides an overview of the Visual Studio tools that you can use to work with textures and images, 3D models, and shader effects.
Image Editor	Describes how to use the Visual Studio Image Editor to work with textures and images.
Model Editor	Describes how to use the Visual Studio Model Editor to work with 3D models.

Shader Designer nodes

10/18/2019 • 3 minutes to read • [Edit Online](#)

The articles in this section of the documentation contain information about the various Shader Designer nodes that you can use to create graphics effects.

Nodes and node types

The Shader Designer represents visual effects as a graph. These graphs are built from nodes that are specifically chosen and connected in precise ways to achieve the intended effect. Each node represents either a piece of information or a mathematical function, and the connections between them represent how the information flows through the graph to produce the result. The Shader Designer provides six different node types—filters, texture nodes, parameters, constants, utility nodes, and math nodes—and several individual nodes belong to each type. These nodes and node types are described in the other articles in this section. For more information, see the links at the end of this document.

Node structure

All nodes are made up of a combination of common elements. Every node has at least one output terminal on its right-hand side (except the final color node, which represents the output of the shader). Nodes that represent calculations or texture samplers have input terminals on their left-hand sides, but nodes that represent information have no input terminals. Output terminals are connected to input terminals to move information from one node to another.

Promotion of inputs

Because the Shader Designer must ultimately generate HLSL source code so that the effect can be used in a game or app, Shader Designer nodes are subject to the type-promotion rules that HLSL uses. Because graphics hardware operates primarily on floating-point values, type promotion between different types—for example, from `int` to `float`, or from `float` to `double`—is uncommon. Instead, because graphics hardware uses the same operation on multiple pieces of information at once, a different kind of promotion can occur in which the shorter of a number of inputs is lengthened to match the size of the longest input. How it is lengthened depends on the type of the input, and also on the operation itself:

- **If the smaller type is a scalar value, then:**

The value of the scalar is replicated into a vector that is equal in size to the larger input. For example, the scalar input 5.0 becomes the vector (5.0, 5.0, 5.0) when the largest input of the operation is a three-element vector, regardless of what the operation is.

- **If the smaller type is a vector, and the operation is multiplicative (*, /, %, and so on), then:**

The value of the vector is copied into the leading elements of a vector that is equal in size to the larger input, and the trailing elements are set to 1.0. For example, the vector input (5.0, 5.0) becomes the vector (5.0, 5.0, 1.0, 1.0) when it's multiplied by a four-element vector. This preserves the third and fourth elements of the output by using the multiplicative identity, 1.0.

- **If the smaller type is a vector, and the operation is additive (+, -, and so on), then:**

The value of the vector is copied into the leading elements of a vector that is equal in size to the larger input, and the trailing elements are set to 0.0. For example, the vector input (5.0, 5.0) becomes the vector (5.0, 5.0, 0.0, 0.0) when it's added to a four-element vector. This preserves the third and fourth elements of the output by using the additive identity, 0.0.

Related topics

TITLE	DESCRIPTION
Constant nodes	Describes nodes that you can use to represent literal values and interpolated vertex-state information in shader calculations. Because vertex-state is interpolated—and therefore, is different for each pixel—each pixel-shader instance receives a different version of the constant.
Parameter nodes	Describes nodes that you can use to represent camera position, material properties, lighting parameters, time, and other app-state information in shader calculations.
Texture nodes	Describes the nodes that you can use to sample various texture types and geometries, and to produce or transform texture coordinates in common ways.
Math nodes	Describes the nodes that you can use to perform algebraic, logic, trigonometric, and other mathematical operations that map directly to HLSL instructions.
Utility nodes	Describes the nodes that you can use to perform common lighting calculations and other common operations that do not map directly to HLSL instructions.
Filter nodes	Describes the nodes that you can use to perform texture filtering and color filtering.

Constant nodes

10/18/2019 • 3 minutes to read • [Edit Online](#)

In the Shader Designer, constant nodes represent literal values and interpolated vertex attributes in pixel-shader calculations. Because vertex attributes are interpolated—and so, are different for each pixel—each pixel-shader instance receives a different version of the constant. This gives each pixel a unique appearance.

Vertex attribute interpolation

The image of a 3D scene in a game or app is made by mathematically transforming a number of objects—which are defined by vertices, vertex attributes, and primitive definitions—into on-screen pixels. All of the information that's required to give a pixel its unique appearance is supplied through vertex attributes, which are blended together according to the pixel's proximity to the different vertices that make up its *primitive*. A primitive is a basic rendering element; that is, a simple shape such as a point, a line, or a triangle. A pixel that's very close to just one of the vertices receives constants that are nearly identical to that vertex, but a pixel that's evenly spaced between all the vertices of a primitive receives constants that are the average of those vertices. In graphics programming, the constants that the pixels receive are said to be *interpolated*. Providing constant data to pixels in this way produces very good visual quality and at the same time reduces memory footprint and bandwidth requirements.

Although each pixel-shader instance receives only one set of constant values and cannot change these values, different pixel-shader instances receive different sets of constant data. This design enables a shader program to produce a different color output for each pixel in the primitive.

Constant node reference

NODE	DETAILS	PROPERTIES
Camera Vector	<p>The vector that extends from the current pixel to the camera in world space.</p> <p>You can use this to calculate reflections in world space.</p> <p>Output</p> <p><code>Output : float3</code></p> <p>The vector from the current pixel to the camera.</p>	None
Color Constant	<p>A constant color value.</p> <p>Output</p> <p><code>Output : float4</code></p> <p>The color value.</p>	<p>Output</p> <p>The color value.</p>

Node	Details	Properties
Constant	<p>A constant scalar value.</p> <p>Output</p> <div style="border: 1px solid black; padding: 2px; display: inline-block;">Output : float</div> <p>The scalar value.</p>	<p>Output The scalar value.</p>
2D Constant	<p>A two-component vector constant.</p> <p>Output</p> <div style="border: 1px solid black; padding: 2px; display: inline-block;">Output : float2</div> <p>The vector value.</p>	<p>Output The vector value.</p>
3D Constant	<p>A three-component vector constant.</p> <p>Output</p> <div style="border: 1px solid black; padding: 2px; display: inline-block;">Output : float3</div> <p>The vector value.</p>	<p>Output The vector value.</p>
4D Constant	<p>A four-component vector constant.</p> <p>Output</p> <div style="border: 1px solid black; padding: 2px; display: inline-block;">Output : float4</div> <p>The color value.</p>	<p>Output The vector value.</p>
Normalized Position	<p>The position of the current pixel, expressed in normalized device coordinates.</p> <p>The x-coordinate and y-coordinate have values in the range of [-1, 1], the z-coordinate has a value in the range of [0, 1], and the w component contains the point depth value in view space; w is not normalized.</p> <p>Output</p> <div style="border: 1px solid black; padding: 2px; display: inline-block;">Output : float4</div> <p>The position of the current pixel.</p>	None
Point Color	<p>The diffuse color of the current pixel, which is a combination of the material diffuse color and vertex color attributes.</p> <p>Output</p> <div style="border: 1px solid black; padding: 2px; display: inline-block;">Output : float4</div> <p>The diffuse color of the current pixel.</p>	None

Node	Details	Properties
Point Depth	<p>The depth of the current pixel in view space.</p> <p>Output</p> <div style="border: 1px solid black; padding: 2px; display: inline-block;">Output : float</div> <p>The depth of the current pixel.</p>	None
Normalized Point Depth	<p>The depth of the current pixel, expressed in normalized device coordinates.</p> <p>The result has a value in the range of [0, 1].</p> <p>Output</p> <div style="border: 1px solid black; padding: 2px; display: inline-block;">Output : float</div> <p>The depth of the current pixel.</p>	None
Screen Position	<p>The position of the current pixel, expressed in screen coordinates.</p> <p>The screen coordinates are based on the current viewport. The x and y components contain the screen coordinates, the z component contains the depth normalized to a range of [0, 1], and the w component contains the depth value in view space.</p> <p>Output</p> <div style="border: 1px solid black; padding: 2px; display: inline-block;">Output : float4</div> <p>The position of the current pixel.</p>	None
Surface Normal	<p>The surface normal of the current pixel in object space.</p> <p>You can use this to calculate lighting contributions and reflections in object space.</p> <p>Output</p> <div style="border: 1px solid black; padding: 2px; display: inline-block;">Output : float3</div> <p>The surface normal of the current pixel.</p>	None

NODE	DETAILS	PROPERTIES
Tangent Space Camera Vector	<p>The vector that extends from the current pixel to the camera in tangent space.</p> <p>You can use this to calculate reflections in tangent space.</p> <p>Output</p> <div data-bbox="595 460 777 489" style="border: 1px solid black; padding: 2px;">Output : float3</div> <p>The vector from the current pixel to the camera.</p>	None
Tangent Space Light Direction	<p>The vector that defines the direction in which light is cast from a light source in the tangent space of the current pixel.</p> <p>You can use this to calculate lighting and specular contributions in tangent space.</p> <p>Output:</p> <div data-bbox="595 920 777 950" style="border: 1px solid black; padding: 2px;">Output : float3</div> <p>The vector from the current pixel to a light source.</p>	None
World Normal	<p>The surface normal of the current pixel in world space.</p> <p>You can use this to calculate lighting contributions and reflections in world space.</p> <p>Output</p> <div data-bbox="595 1347 777 1376" style="border: 1px solid black; padding: 2px;">Output : float3</div> <p>The surface normal of the current pixel.</p>	None
World Position	<p>The position of the current pixel in world space.</p> <p>Output</p> <div data-bbox="595 1628 777 1657" style="border: 1px solid black; padding: 2px;">Output : float4</div> <p>The position of the current pixel.</p>	None

Parameter nodes

10/18/2019 • 3 minutes to read • [Edit Online](#)

In the Shader Designer, parameter nodes represent inputs to the shader that are under the control of the app on a per-draw basis, for example, material properties, directional lights, camera position, and time. Because you can change these parameters with each draw call, you can use the same shader to give an object different appearances.

Parameter node reference

NODE	DETAILS	PROPERTIES
Camera World Position	<p>The position of the camera in world space.</p> <p>Output:</p> <div style="border: 1px solid #ccc; padding: 2px;"><code>Output : float4</code></div> <p>The position of the camera.</p>	None
Light Direction	<p>The vector that defines the direction in which light is cast from a light source in world space.</p> <p>You can use this to calculate lighting and specular contributions in world space.</p> <p>Output:</p> <div style="border: 1px solid #ccc; padding: 2px;"><code>Output : float3</code></div> <p>The vector from the current pixel to a light source.</p>	None
Material Ambient	<p>The diffuse color contribution of the current pixel that is attributed to indirect lighting.</p> <p>The diffuse color of a pixel simulates how lighting interacts with rough surfaces. You can use the Material Ambient parameter to approximate how indirect lighting contributes to the appearance of an object in the real world.</p> <p>Output:</p> <div style="border: 1px solid #ccc; padding: 2px;"><code>Output : float4</code></div> <p>The diffuse color of the current pixel that's due to indirect—that is, ambient—lighting.</p>	<p>Access</p> <p>Public to enable the property to be set from the Model Editor; otherwise, Private.</p> <p>Value</p> <p>The diffuse color of the current pixel that's due to indirect—that is, ambient—lighting.</p>

NODE	DETAILS	PROPERTIES
Material Diffuse	<p>A color that describes how the current pixel diffuses direct lighting.</p> <p>The diffuse color of a pixel simulates how lighting interacts with rough surfaces. You can use the Material Diffuse parameter to change how the current pixel diffuses direct lighting—that is, directional, point, and spot lights.</p> <p>Output:</p> <div style="border: 1px solid #ccc; padding: 2px; display: inline-block;">Output : float4</div> <p>A color that describes how the current pixel diffuses direct lighting.</p>	<p>Access Public to enable the property to be set from the Model Editor; otherwise, Private.</p> <p>Value A color that describes how the current pixel diffuses direct lighting.</p>
Material Emissive	<p>The color contribution of the current pixel that is attributed to lighting that it supplies to itself.</p> <p>You can use this to simulate a glowing object; that is, an object that supplies its own light. This light doesn't affect other objects.</p> <p>Output:</p> <div style="border: 1px solid #ccc; padding: 2px; display: inline-block;">Output : float4</div> <p>The color contribution of the current pixel that's due to self-provided lighting.</p>	<p>Access Public to enable the property to be set from the Model Editor; otherwise, Private.</p> <p>Value The color contribution of the current pixel that's due to self-provided lighting.</p>
Material Specular	<p>A color that describes how the current pixel reflects direct lighting.</p> <p>The specular color of a pixel simulates how lighting interacts with smooth, mirror-like surfaces. You can use the Material Specular parameter to change how the current pixel reflects direct lighting—that is, directional, point, and spot lights.</p> <p>Output:</p> <div style="border: 1px solid #ccc; padding: 2px; display: inline-block;">Output : float4</div> <p>A color that describes how the current pixel reflects direct lighting.</p>	<p>Access Public to allow the property to be set from the Model Editor; otherwise, Private.</p> <p>Value A color that describes how the current pixel reflects direct lighting.</p>

NODE	DETAILS	PROPERTIES
Material Specular Power	<p>A scalar value that describes the intensity of specular highlights.</p> <p>The larger the specular power, the more intense and far-reaching the specular highlights become.</p> <p>Output:</p> <div style="border: 1px solid black; padding: 2px; display: inline-block;">Output : float</div> <p>An exponential term that describes the intensity of specular highlights on the current pixel.</p>	<p>Access Public to enable the property to be set from the Model Editor; otherwise, Private.</p> <p>Value The exponent that defines the intensity of specular highlights on the current pixel.</p>
Normalized Time	<p>The time in seconds, normalized to the range [0, 1], such that when time reaches 1, it resets to 0.</p> <p>You can use this as a parameter in shader calculations, for example, to animate texture coordinates, color values, or other attributes.</p> <p>Output:</p> <div style="border: 1px solid black; padding: 2px; display: inline-block;">Output : float</div> <p>The normalized time, in seconds.</p>	None
Time	<p>The time in seconds.</p> <p>You can use this as a parameter in shader calculations, for example, to animate texture coordinates, color values, or other attributes.</p> <p>Output:</p> <div style="border: 1px solid black; padding: 2px; display: inline-block;">Output : float</div> <p>The time, in seconds.</p>	None

Texture nodes

10/18/2019 • 3 minutes to read • [Edit Online](#)

In the Shader Designer, texture nodes sample various texture types and geometries, and produce or transform texture coordinates. Textures provide color and lighting detail on objects.

Texture node reference

NODE	DETAILS	PROPERTIES
Cubemap Sample	<p>Takes a color sample from a cubemap at the specified coordinates.</p> <p>You can use a cubemap to provide color detail for reflection effects, or to apply to a spherical object a texture that has less distortion than a 2D texture.</p> <p>Input:</p> <p><code>UVW : float3</code></p> <p>A vector that specifies the location on the texture cube where the sample is taken. The sample is taken where this vector intersects the cube.</p> <p>Output:</p> <p><code>Output : float4</code></p> <p>The color sample.</p>	<p>Texture The texture register that's associated with the sampler.</p>
Normal Map Sample	<p>Takes a normal sample from a 2D normal map at the specified coordinates</p> <p>You can use a normal map to simulate the appearance of additional geometric detail on the surface of an object. Normal maps contain packed data that represents a unit vector instead of color data</p> <p>Input:</p> <p><code>UV : float2</code></p> <p>The coordinates where the sample is taken.</p> <p>Output:</p> <p><code>Output : float3</code></p> <p>The normal sample.</p>	<p>Axis Adjustment The factor that's used to adjust the handedness of the normal map sample.</p> <p>Texture The texture register that's associated with the sampler.</p>

NODE	DETAILS	PROPERTIES
Pan UV	<p>Pans the specified texture coordinates as a function of time.</p> <p>You can use this to move a texture or normal map across the surface of an object.</p> <p>Input:</p> <p><code>UV : float2</code> The coordinates to pan.</p> <p><code>Time : float</code> The length of time to pan by, in seconds.</p> <p>Output:</p> <p><code>Output : float2</code> The panned coordinates.</p>	<p>Speed X The number of texels that are panned along the x axis, per second.</p> <p>Speed Y The number of texels that are panned along the y axis, per second.</p>
Parallax UV	<p>Displaces the specified texture coordinates as a function of height and viewing angle.</p> <p>The effect this creates is known as <i>parallax mapping</i>, or virtual displacement mapping. You can use it to create an illusion of depth on a flat surface.</p> <p>Input:</p> <p><code>UV : float2</code> The coordinates to displace.</p> <p><code>Height : float</code> The heightmap value that's associated with the <code>UV</code> coordinates.</p> <p>Output:</p> <p><code>Output : float2</code> The displaced coordinates.</p>	<p>Depth Plane The reference depth for the parallax effect. By default, the value is 0.5. Smaller values lift the texture; larger values sink it into the surface.</p> <p>Depth Scale The scale of the parallax effect. This makes the apparent depth more or less pronounced. Typical values range from 0.02 to 0.1.</p>

Node	Details	Properties
Rotate UV	<p>Rotates the specified texture coordinates around a central point as a function of time.</p> <p>You can use this to spin a texture or normal map on the surface of an object.</p> <p>Input:</p> <ul style="list-style-type: none"> UV : float2 The coordinates to rotate. Time : float The length of time to pan by, in seconds. <p>Output:</p> <ul style="list-style-type: none"> Output : float2 The rotated coordinates. 	<p>Center X The x coordinate that defines the center of rotation.</p> <p>Center Y The y coordinate that defines the center of rotation.</p> <p>Speed The angle, in radians, by which the texture rotates per second.</p>
Texture Coordinate	<p>The texture coordinates of the current pixel.</p> <p>The texture coordinates are determined by interpolating among the texture coordinate attributes of nearby vertices. You can think of this as the position of the current pixel in texture space.</p> <p>Output:</p> <ul style="list-style-type: none"> Output : float2 The texture coordinates. 	None
Texture Dimensions	<p>Outputs the width and height of a 2D texture map.</p> <p>You can use the texture dimensions to consider the width and height of the texture in a shader.</p> <p>Output:</p> <ul style="list-style-type: none"> Output : float2 The width and height of the texture, expressed as a vector. The width is stored in the first element of the vector. The height is stored in the second element. 	<p>Texture The texture register that's associated with the texture dimensions.</p>

NODE	DETAILS	PROPERTIES
Texel Delta	<p>Outputs the delta (distance) between the texels of a 2D texture map.</p> <p>You can use the texel delta to sample neighboring texel values in a shader.</p> <p>Output:</p> <div style="border: 1px solid black; padding: 2px; display: inline-block;">Output : float2</div> <p>The delta (distance) from a texel to the next texel (moving diagonally in the positive direction), expressed as a vector in normalized texture space. You can derive the positions of all neighboring texels by selectively ignoring or negating the U or V coordinates of the delta.</p>	<p>Texture The texture register that's associated with the texel delta.</p>
Texture Sample	<p>Takes a color sample from a 2D texture map at the specified coordinates.</p> <p>You can use a texture map to provide color detail on the surface of an object.</p> <p>Input:</p> <div style="border: 1px solid black; padding: 2px; display: inline-block;">UV : float2</div> <p>The coordinates where the sample is taken.</p> <p>Output:</p> <div style="border: 1px solid black; padding: 2px; display: inline-block;">Output : float4</div> <p>The color sample.</p>	<p>Texture The texture register that's associated with the sampler.</p>

Math nodes

10/18/2019 • 9 minutes to read • [Edit Online](#)

In the Shader Designer, math nodes perform algebraic, logic, trigonometric, and other mathematical operations.

NOTE

When you work with math nodes in the Shader Designer, type promotion is especially evident. To learn how type promotion affects input parameters, see the "Promotion of inputs" section in [Shader Designer nodes](#).

Math node reference

NODE	DETAILS	PROPERTIES
Abs	<p>Computes the absolute value of the specified input per component.</p> <p>For each component of input <code>x</code>, negative values are made positive so that every component of the result has a positive value.</p> <p>Input:</p> <p><code>x</code> : <code>float</code>, <code>float2</code>, <code>float3</code>, or <code>float4</code></p> <p>The values for which to determine the absolute value.</p> <p>Output:</p> <p><code>Output</code> : same as input <code>x</code></p> <p>The absolute value, per component.</p>	None
Add	<p>Computes the component-wise sum of the specified inputs per component.</p> <p>For each component of the result, the corresponding components of input <code>x</code> and input <code>y</code> are added together.</p> <p>Input:</p> <p><code>x</code> : <code>float</code>, <code>float2</code>, <code>float3</code>, or <code>float4</code></p> <p>One of the values to add together.</p> <p><code>y</code> : same as input <code>x</code></p> <p>One of the values to add together.</p> <p>Output:</p> <p><code>Output</code> : same as input <code>x</code></p> <p>The sum, per component.</p>	None

NODE	DETAILS	PROPERTIES
Ceil	<p>Computes the ceiling of the specified input per component.</p> <p>The ceiling of a value is the smallest integer that's greater than or equal to that value.</p> <p>Input:</p> <div style="border: 1px solid #ccc; padding: 2px;"><code>X : float, float2, float3, or float4</code></div> <p>The values for which to compute the ceiling.</p> <p>Output:</p> <div style="border: 1px solid #ccc; padding: 2px;"><code>Output : same as input X</code></div> <p>The ceiling, per component.</p>	None
Clamp	<p>Clamps each component of the specified input to a predefined range.</p> <p>For each component of the result, values that are below the defined range are made equal to the minimum value in the range, values that are above the defined range are made equal to the maximum value in the range, and values that are in the range are not changed.</p> <p>Input:</p> <div style="border: 1px solid #ccc; padding: 2px;"><code>X : float, float2, float3, or float4</code></div> <p>The values to clamp.</p> <p>Output:</p> <div style="border: 1px solid #ccc; padding: 2px;"><code>Output : same as input X</code></div> <p>The clamped value, per component.</p>	<p>Max The largest possible value in the range.</p> <p>Min The smallest possible value in the range.</p>

NODE	DETAILS	PROPERTIES
Cos	<p>Computes the cosine of the specified input, in radians, per component.</p> <p>For each component of the result, the cosine of the corresponding component, which is provided in radians, is calculated. The result has components that have values in the range of [-1, 1].</p> <p>Input:</p> <div style="border: 1px solid #ccc; padding: 2px;"><code>x : float , float2 , float3 , or float4</code></div> <p>The values to compute the cosine of, in radians.</p> <p>Output:</p> <div style="border: 1px solid #ccc; padding: 2px;"><code>Output : same as input x</code></div> <p>The cosine, per component.</p>	None
Cross	<p>Computes the cross product of the specified three-component vectors.</p> <p>You can use the cross product to compute the normal of a surface that's defined by two vectors.</p> <p>Input:</p> <div style="border: 1px solid #ccc; padding: 2px;"><code>X : float3</code></div> <p>The vector on the left-hand-side of the cross product.</p> <div style="border: 1px solid #ccc; padding: 2px;"><code>Y : float3</code></div> <p>The vector on the right-hand-side of the cross product.</p> <p>Output:</p> <div style="border: 1px solid #ccc; padding: 2px;"><code>Output : float3</code></div> <p>The cross product.</p>	None

NODE	DETAILS	PROPERTIES
Distance	<p>Computes the distance between the specified points.</p> <p>The result is a positive scalar value.</p> <p>Input:</p> <p><code>x</code> : float , float2 , float3 , or float4 One of the points to determine the distance between.</p> <p><code>y</code> : same as input <code>x</code> One of the points to determine the distance between.</p> <p>Output:</p> <p><code>Output</code> : same as input <code>x</code> The distance.</p>	None
Divide	<p>Computes the component-wise quotient of the specified inputs.</p> <p>For each component of the result, the corresponding component of input <code>x</code> is divided by the corresponding component of input <code>y</code>.</p> <p>Input:</p> <p><code>x</code> : float , float2 , float3 , or float4 The dividend values.</p> <p><code>y</code> : same as input <code>x</code> The divisor values.</p> <p>Output:</p> <p><code>Output</code> : same as input <code>x</code> The quotient, per component.</p>	None

NODE	DETAILS	PROPERTIES
Dot	<p>Computes the dot product of the specified vectors.</p> <p>The result is a scalar value. You can use the dot product to determine the angle between two vectors.</p> <p>Input:</p> <p><code>x</code> : float , float2 , float3 , or float4 One of the terms.</p> <p><code>y</code> : same as input <code>x</code> One of the terms.</p> <p>Output:</p> <p><code>Output</code> : float The dot product.</p>	None
Floor	<p>Computes the floor of the specified input per component.</p> <p>For each component of the result, its value is the largest whole integer value that's less than or equal to the corresponding component of the input. Every component of the result is a whole integer.</p> <p>Input:</p> <p><code>x</code> : float , float2 , float3 , or float4 The values for which to compute the floor.</p> <p>Output:</p> <p><code>Output</code> : same as input <code>x</code> The floor, per component.</p>	None

NODE	DETAILS	PROPERTIES
Fmod	<p>Computes the component-wise modulus (remainder) of the specified inputs.</p> <p>For each component of the result, some integral (whole-number) multiple, m, of the corresponding component of input y is subtracted from the corresponding component of input x, leaving a remainder. The multiple, m, is chosen such that the remainder is less than the corresponding component of input y and has the same sign as the corresponding component of input x. For example, $\text{fmod}(-3.14, 1.5)$ is -0.14.</p> <p>Input:</p> <p>x : float , float2 , float3 , or float4 The dividend values.</p> <p>y : same as input x The divisor values.</p> <p>Output:</p> <p>Output : same as input x The modulus, per component.</p>	None
Frac	<p>Removes the integral (whole-number) part of the specified input per component.</p> <p>For each component of the result, the integral part of the corresponding component of the input is removed, but the fractional part and sign are retained. This fractional value falls in the range [0, 1). For example, the value -3.14 becomes the value -0.14.</p> <p>Input:</p> <p>x : float , float2 , float3 , or float4 The values for which to compute the fractional part.</p> <p>Output:</p> <p>Output : same as input x The fractional part, per component.</p>	None

NODE	DETAILS	PROPERTIES
Lerp	<p>Linear Interpolation. Computes the component-wise weighted average of the specified inputs.</p> <p>For each component of the result, the weighted average of the corresponding components of the inputs <code>x</code> and <code>y</code>. The weight is provided by <code>Percent</code>, a scalar, and is uniformly applied to all components. You can use this to interpolate between points, colors, attributes, and other values.</p> <p>Input:</p> <p><code>x</code> : <code>float</code>, <code>float2</code>, <code>float3</code>, or <code>float4</code></p> <p>The originating value. When <code>Percent</code> is zero, the result is equal to this input.</p> <p><code>y</code> : same as input <code>x</code></p> <p>The terminal value. When <code>Percent</code> is one, the result is equal to this input.</p> <p><code>Percent</code> : <code>float</code></p> <p>A scalar weight that's expressed as a percentage of the distance from input <code>x</code> towards input <code>y</code>.</p> <p>Output:</p> <p><code>Output</code> : same as input <code>x</code></p> <p>A value that's collinear with the specified inputs.</p>	None

Node	Details	Properties
Multiply Add	<p>Computes the component-wise multiply-add of the specified inputs.</p> <p>For each component of the result, the product of the corresponding components of the inputs M and A is added to the corresponding component of input B. This operation sequence is found in common formulas—for example, in the point-slope formula of a line, and in the formula to scale and then bias an input.</p> <p>Input:</p> <p>M : float , float2 , float3 , or float4</p> <p>One of the values to multiply together.</p> <p>A : same as input M</p> <p>One of the values to multiply together.</p> <p>B : same as input M</p> <p>The values to add to the product of the other two inputs.</p> <p>Output:</p> <p>Output : same as input M</p> <p>The result of the multiply-add, per component.</p>	None
Max	<p>Computes the component-wise maximum of the specified inputs.</p> <p>For each component of the result, the greater of the corresponding components of the inputs is taken.</p> <p>Input:</p> <p>X : float , float2 , float3 , or float4</p> <p>One of the values for which to compute the maximum.</p> <p>Y : same as input X</p> <p>One of the values for which to compute the maximum.</p> <p>Output:</p> <p>Output : same as input X</p> <p>The maximum value, per component.</p>	None

NODE	DETAILS	PROPERTIES
Min	<p>Computes the component-wise minimum of the specified inputs.</p> <p>For each component of the result, the lesser of the corresponding components of the inputs is taken.</p> <p>Input:</p> <p><code>x</code> : float, float2, float3, or float4 <code>y</code> : same as input <code>x</code></p> <p>One of the values for which to compute the minimum.</p> <p><code>y</code> : same as input <code>x</code> <code>Output</code> : same as input <code>x</code></p> <p>One of the values for which to compute the minimum.</p> <p>Output:</p> <p><code>Output</code> : same as input <code>x</code> <code>Output</code> : same as input <code>x</code></p> <p>The minimum value, per component.</p>	None
Multiply	<p>Computes the component-wise product of the specified inputs.</p> <p>For each component of the result, the corresponding components of the inputs <code>x</code> and <code>y</code> are multiplied together.</p> <p>Input:</p> <p><code>x</code> : float, float2, float3, or float4 <code>y</code> : same as input <code>x</code></p> <p>One of the values to multiply together.</p> <p><code>y</code> : same as input <code>x</code> <code>Output</code> : same as input <code>x</code></p> <p>One of the values to multiply together.</p> <p>Output:</p> <p><code>Output</code> : same as input <code>x</code> <code>Output</code> : same as input <code>x</code></p> <p>The product, per component.</p>	None

NODE	DETAILS	PROPERTIES
Normalize	<p>Normalizes the specified vector.</p> <p>A normalized vector retains the direction of the original vector, but not its magnitude. You can use normalized vectors to simplify calculations where the magnitude of a vector is not important.</p> <p>Input:</p> <p><code>x</code> : <code>float2</code>, <code>float3</code>, or <code>float4</code> The vector to normalize.</p> <p>Output:</p> <p><code>Output</code> : same as input <code>x</code> The normalized vector.</p>	None
One Minus	<p>Computes the difference between 1 and the specified input per component.</p> <p>For each component of the result, the corresponding component of the input is subtracted from 1.</p> <p>Input:</p> <p><code>x</code> : <code>float</code>, <code>float2</code>, <code>float3</code>, or <code>float4</code> The values to be subtracted from 1.</p> <p>Output:</p> <p><code>Output</code> : same as input <code>x</code> The difference between 1 and the specified input, per component.</p>	None

NODE	DETAILS	PROPERTIES
Power	<p>Computes the component-wise exponentiation (power) of the specified inputs.</p> <p>For each component of the result, the corresponding component of input x is raised to the power of the corresponding component of the input y.</p> <p>Input:</p> <p>x : float , float2 , float3 , or float4 The base values</p> <p>y : same as input x The exponent values.</p> <p>Output:</p> <p>Output : same as input x The exponentiation, per component.</p>	None
Saturate	<p>Clamps each component of the specified input to the range [0, 1].</p> <p>You can use this range to represent percentages and other relative measurements in calculations. For each component of the result, the corresponding component values of the input that are less than 0 are made equal to 0, values that are larger than 1 are made equal to 1, and values that are in the range are not changed.</p> <p>Input:</p> <p>x : float , float2 , float3 , or float4 The values to saturate.</p> <p>Output:</p> <p>Output : same as input x The saturated value, per component.</p>	None

NODE	DETAILS	PROPERTIES
Sin	<p>Computes the sine of the specified input, in radians, per component.</p> <p>For each component of the result, the sine of the corresponding component, which is provided in radians, is calculated. The result has components that have values in the range [-1, 1].</p> <p>Input:</p> <div style="border: 1px solid #ccc; padding: 2px;"><code>x : float, float2, float3, or float4</code></div> <p>The values to compute the sine of, in radians.</p> <p>Output:</p> <div style="border: 1px solid #ccc; padding: 2px;"><code>output : same as input x</code></div> <p>The sine, per component.</p>	None
Sqrt	<p>Computes the square root of the specified input, per component.</p> <p>For each component of the result, the square root of the corresponding component is calculated.</p> <p>Input:</p> <div style="border: 1px solid #ccc; padding: 2px;"><code>x : float, float2, float3, or float4</code></div> <p>The values for which to compute the square root.</p> <p>Output:</p> <div style="border: 1px solid #ccc; padding: 2px;"><code>output : same as input x</code></div> <p>The square root, per component.</p>	None

NODE	DETAILS	PROPERTIES
Subtract	<p>Computes the component-wise difference of the specified inputs.</p> <p>For each component of the result, the corresponding component of input <code>y</code> is subtracted from the corresponding component of input <code>x</code>. You can use this to compute the vector that extends from the first input to the second.</p> <p>Input:</p> <p><code>x</code> : <code>float</code>, <code>float2</code>, <code>float3</code>, or <code>float4</code> The values to be subtracted from.</p> <p><code>y</code> : same as input <code>x</code> The values to subtract from input <code>x</code>.</p> <p>Output:</p> <p><code>Output</code> : same as input <code>x</code> The difference, per component.</p>	None
Transform 3D Vector	<p>Transforms the specified 3D vector into a different space.</p> <p>You can use this to bring points or vectors into a common space so that you can use them to perform meaningful calculations.</p> <p>Input:</p> <p><code>Vector</code> : <code>float3</code> The vector to transform.</p> <p>Output:</p> <p><code>Output</code> : <code>float3</code> The transformed vector.</p>	<p>From System The native space of the vector.</p> <p>To System The space to transform the vector into.</p>

Utility nodes

10/18/2019 • 3 minutes to read • [Edit Online](#)

In the Shader Designer, utility nodes represent common, useful shader calculations that don't fit neatly into the other categories. Some utility nodes perform simple operations such as appending vectors together or choosing results conditionally, and others perform complex operations such as computing lighting contributions according to popular lighting models.

Utility node reference

NODE	DETAILS	PROPERTIES
Append Vector	<p>Creates a vector by appending the specified inputs together.</p> <p>Input:</p> <p><code>Vector</code> : <code>float</code>, <code>float2</code>, or <code>float3</code> The values to append to.</p> <p><code>Value to Append</code> : <code>float</code> The value to append.</p> <p>Output:</p> <p><code>Output</code> : <code>float2</code>, <code>float3</code>, or <code>float4</code> depending on the type of input <code>Vector</code> The new vector.</p>	None

NODE	DETAILS	PROPERTIES
Fresnel	<p>Computes the Fresnel fall-off based on the specified surface normal.</p> <p>The Fresnel fall-off value expresses how closely the surface normal of the current pixel coincides with the view vector. When the vectors are aligned, the result of the function is 0; the result increases as the vectors become less similar, and reaches its maximum when the vectors are orthogonal. You can use this to make an effect more or less apparent based on the relationship between the orientation of the current pixel and the camera.</p> <p>Input:</p> <p><code>Surface Normal : float3</code> The surface normal of the current pixel, defined in the current pixel's tangent space. You can use this to perturb the apparent surface normal, as in normal mapping.</p> <p>Output:</p> <p><code>Output : float</code> The reflectivity of the current pixel.</p>	<p>Exponent The exponent that's used to calculate the Fresnel fall-off.</p>

NODE	DETAILS	PROPERTIES
If	<p>Conditionally chooses one of three potential results per component. The condition is defined by the relationship between two other specified inputs.</p> <p>For each component of the result, the corresponding component of one of the three potential results is chosen, based on the relationship between the corresponding components of the first two inputs.</p> <p>Input:</p> <p><code>x</code> : float , float2 , float3 , or float4 The left-hand side value to compare.</p> <p><code>y</code> : same type as input <code>x</code> The right-hand side value to compare.</p> <p><code>x > y</code> : same type as input <code>x</code> The values that are chosen when <code>x</code> is greater than <code>y</code>.</p> <p><code>x = y</code> : same type as input <code>x</code> The values that are chosen when <code>x</code> is equal to <code>y</code>.</p> <p><code>x < y</code> : same type as input <code>x</code> The values that are chosen when <code>x</code> is less than <code>y</code>.</p> <p>Output:</p> <p><code>Output</code> : float3 The chosen result, per component.</p>	None

Node	Details	Properties
Lambert	<p>Computes the color of the current pixel according to the Lambert lighting model, by using the specified surface normal.</p> <p>This color is the sum of ambient color and diffuse lighting contributions under direct lighting. Ambient color approximates the total contribution of indirect lighting, but looks flat and dull without the help of additional lighting. Diffuse lighting helps add shape and depth to an object.</p> <p>Input:</p> <p><code>Surface Normal : float3</code> The surface normal of the current pixel, defined in the current pixel's tangent space. You can use this to perturb the apparent surface normal, as in normal mapping.</p> <p><code>Diffuse Color : float3</code> The diffuse color of the current pixel, typically the Point Color. If no input is provided, the default value is white.</p> <p>Output:</p> <p><code>Output : float3</code> The diffuse color of the current pixel.</p>	None
Mask Vector	<p>Masks components of the specified vector.</p> <p>You can use this to remove specific color channels from a color value, or to prevent specific components from having an effect on subsequent calculations.</p> <p>Input:</p> <p><code>Vector : float4</code> The vector to mask.</p> <p>Output:</p> <p><code>Output : float4</code> The masked vector.</p>	<p>Red / X False to mask out the red (x) component; otherwise, True.</p> <p>Green / Y False to mask out the green (y) component; otherwise, True.</p> <p>Blue / Z False to mask out the blue (z) component; otherwise, True.</p> <p>Alpha / W False to mask out the alpha (w) component; otherwise, True.</p>

NODE	DETAILS	PROPERTIES
Reflection Vector	<p>Computes the reflection vector for the current pixel in tangent space, based on the camera position.</p> <p>You can use this to calculate reflections, cubemap coordinates, and specular lighting contributions</p> <p>Input:</p> <div style="border: 1px solid black; padding: 2px; display: inline-block;">Tangent Space Surface Normal :</div> <div style="border: 1px solid black; padding: 2px; display: inline-block;">float3</div> <p>The surface normal of the current pixel, defined in the current pixel's tangent space. You can use this to perturb the apparent surface normal, as in normal mapping.</p> <p>Output:</p> <div style="border: 1px solid black; padding: 2px; display: inline-block;">Output : float3</div> <p>The reflection vector.</p>	None
Specular	<p>Computes the specular lighting contribution according to the Phong lighting model, by using the specified surface normal.</p> <p>Specular lighting gives a shiny, reflective appearance to an object, for example, water, plastic, or metals.</p> <p>Input:</p> <div style="border: 1px solid black; padding: 2px; display: inline-block;">Surface Normal : float3</div> <p>The surface normal of the current pixel, defined in the current pixel's tangent space. You can use this to perturb the apparent surface normal, as in normal mapping.</p> <p>Output:</p> <div style="border: 1px solid black; padding: 2px; display: inline-block;">Output : float3</div> <p>The color contribution of specular highlights.</p>	None

Filter nodes

10/18/2019 • 2 minutes to read • [Edit Online](#)

In the Shader Designer, filter nodes transform an input—for example, a color or texture sample—into a figurative color value. These figurative color values are commonly used in non-photorealistic rendering or as components in other visual effects.

Filter node reference

Node	Details	Properties
Blur	<p>Blurs pixels in a texture by using a Gaussian function.</p> <p>You can use this to reduce color detail or noise in a texture.</p> <p>Input:</p> <p><code>UV : float2</code></p> <p>The coordinates of the texel to test.</p> <p>Output:</p> <p><code>Output : float4</code></p> <p>The blurred color value.</p>	<p>Texture</p> <p>The texture register that's associated with the sampler that's used during blurring.</p>
Desaturate	<p>Reduces the amount of color in the specified color.</p> <p>As color is removed, the color value approaches its gray-scale equivalent.</p> <p>Input:</p> <p><code>RGB : float3</code></p> <p>The color to desaturate.</p> <p><code>Percent : float</code></p> <p>The percent of color to remove, expressed as a normalized value in the range [0, 1].</p> <p>Output:</p> <p><code>Output : float3</code></p> <p>The desaturated color.</p>	<p>Luminance</p> <p>The weights that are given to the red, green, and blue color components.</p>

NODE	DETAILS	PROPERTIES
Edge Detection	<p>Detects edges in a texture by using a Canny edge detector. Edge pixels are output as white; non-edge pixels are output as black.</p> <p>You can use this to identify edges in a texture so that you can use additional effects to treat edge pixels.</p> <p>Input:</p> <p>UV : float2 The coordinates of the texel to test.</p> <p>Output:</p> <p>Output : float4 White if the texel is on an edge; otherwise, black.</p>	<p>Texture The texture register that's associated with the sampler that's used during edge detection.</p>
Sharpen	<p>Sharpens a texture.</p> <p>You can use this to highlight fine details in a texture.</p> <p>Input:</p> <p>UV : float2 The coordinates of the texel to test.</p> <p>Output:</p> <p>Output : float4 The blurred color value.</p>	<p>Texture The texture register that's associated with the sampler that's used during sharpening.</p>

How to: Create a basic color shader

10/18/2019 • 2 minutes to read • [Edit Online](#)

This article demonstrates how to use the Shader Designer and the Directed Graph Shader Language (DGSL) to create a flat color shader. This shader sets the final color to a constant RGB color value.

Create a flat color shader

You can implement a flat color shader by writing the color value of an RGB color constant to the final output color.

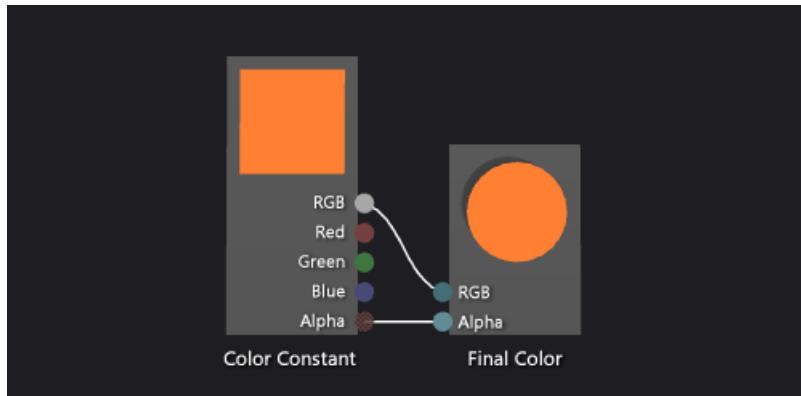
Before you begin, make sure that the **Properties** window and the **Toolbox** are displayed.

1. Create a DGSL shader to work with. For information about how to add a DGSL shader to your project, see the Getting Started section in [Shader Designer](#).
2. Delete the **Point Color** node. Use the **Select** tool to select the **Point Color** node, and then on the menu bar, choose **Edit > Delete**.
3. Add a **Color Constant** node to the graph. In the **Toolbox**, under **Constants**, select **Color Constant** and move it to the design surface.
4. Specify a color value for the **Color Constant** node. Use the **Select** tool to select the **Color Constant** node, and then, in the **Properties** window, in the **Output** property, specify a color value. For orange, specify a value of (1.0, 0.5, 0.2, 1.0).
5. Connect the color constant to the final color. To create the connections, move the **RGB** terminal of the **Color Constant** node to the **RGB** terminal of the **Final Color** node, and then move the **Alpha** terminal of the **Color Constant** node to the **Alpha** terminal of the **Final Color** node. These connections set the final color to the color constant defined in the previous step.

The following illustration shows the completed shader graph and a preview of the shader applied to a cube.

NOTE

In the illustration, an orange color was specified to better demonstrate the effect of the shader.



Certain shapes might provide better previews for some shaders. For more information about how to preview shaders in the Shader Designer, see [Shader Designer](#).

See also

- [How to: Apply a shader to a 3D model](#)
- [How to: Export a shader](#)
- [Shader Designer](#)
- [Shader Designer nodes](#)

How to: Create a basic Lambert shader

10/18/2019 • 2 minutes to read • [Edit Online](#)

This article demonstrates how to use the Shader Designer and the Directed Graph Shader Language (DGSL) to create a lighting shader that implements the classic Lambert lighting model.

The Lambert lighting model

The Lambert lighting model incorporates ambient and directional lighting to shade objects in a 3D scene. The ambient components provide a base level of illumination in the 3D scene. The directional components provide additional illumination from directional (far-away) light sources. Ambient illumination affects all surfaces in the scene equally, regardless of their orientation. For a given surface, it's a product of the ambient color of the surface and the color and intensity of ambient lighting in the scene. Directional lighting affects every surface in the scene differently, based on the orientation of the surface with respect to the direction of the light source. It's a product of the diffuse color and orientation of the surface, and the color, intensity, and direction of the light sources. Surfaces that face directly toward the light source receive the maximum contribution and surfaces that face directly away receive no contribution. Under the Lambert lighting model, the ambient component and one or more directional components are combined to determine the total diffuse color contribution for each point on the object.

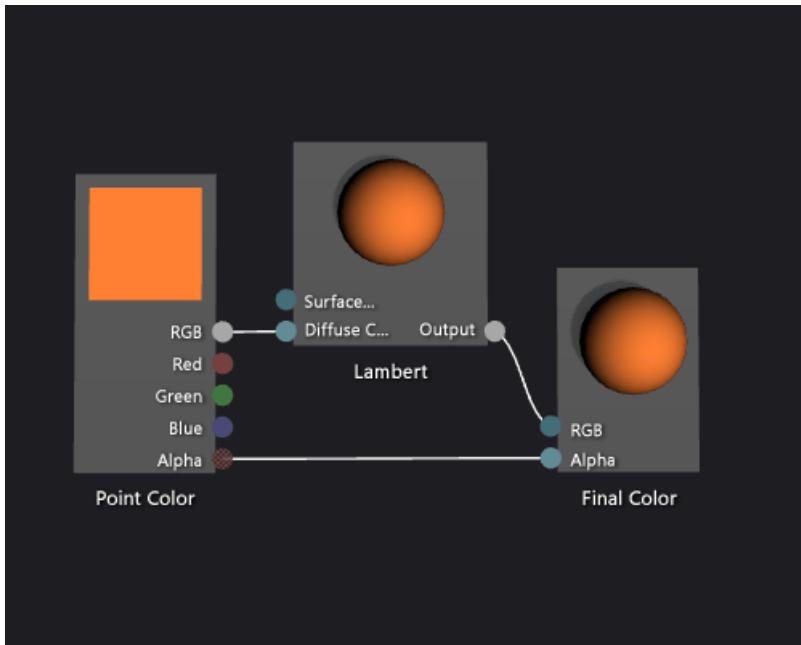
Before you begin, make sure that the **Properties** window and the **Toolbox** are displayed.

1. Create a DGSL shader with which to work. For information about how to add a DGSL shader to your project, see the Getting Started section in [Shader Designer](#).
2. Disconnect the **Point Color** node from the **Final Color** node. Choose the **RGB** terminal of the **Point Color** node, and then choose **Break Links**. Leave the **Alpha** terminal connected.
3. Add a **Lambert** node to the graph. In the **Toolbox**, under **Utility**, select **Lambert** and move it to the design surface. The lambert node computes the total diffuse color contribution of the pixel, based on ambient and diffuse lighting parameters.
4. Connect the **Point Color** node to the **Lambert** node. In **Select** mode, move the **RGB** terminal of the **Point Color** node to the **Diffuse Color** terminal of the **Lambert** node. This connection provides the lambert node with the interpolated diffuse color of the pixel.
5. Connect the computed color value to the final color. Move the **Output** terminal of the **Lambert** node to the **RGB** terminal of the **Final Color** node.

The following illustration shows the completed shader graph and a preview of the shader applied to a teapot model.

NOTE

To better demonstrate the effect of the shader in this illustration, an orange color has been specified by using the **MaterialDiffuse** parameter of the shader. A game or app can use this parameter to supply a unique color value for each object. For information about material parameters, see the Previewing Shaders section in [Shader Designer](#).



Certain shapes might provide better previews for some shaders. For more information about how to preview shaders in the Shader Designer, see the Previewing Shaders section in [Shader Designer](#).

The following illustration shows the shader that's described in this document applied to a 3D model.



For more information about how to apply a shader to a 3D model, see [How to: Apply a Shader to a 3D Model](#).

See also

- [How to: Apply a Shader to a 3D Model](#)
- [How to: Export a Shader](#)
- [How to: Create a Basic Phong Shader](#)
- [Shader Designer](#)
- [Shader Designer Nodes](#)

How to: Create a basic Phong shader

10/18/2019 • 3 minutes to read • [Edit Online](#)

This article demonstrates how to use the Shader Designer and the Directed Graph Shader Language (DGSL) to create a lighting shader that implements the classic Phong lighting model.

The Phong lighting model

The Phong lighting model extends the Lambert lighting model to include specular highlighting, which simulates the reflective properties of a surface. The specular component provides additional illumination from the same directional light sources that are used in the Lambert lighting model, but its contribution to the final color is processed differently. Specular highlighting affects every surface in the scene differently, based on the relationship between the view direction, the direction of the light sources, and the orientation of the surface. It's a product of the specular color, specular power, and orientation of the surface, and the color, intensity, and direction of the light sources. Surfaces that reflect the light source directly at the viewer receive the maximum specular contribution and surfaces that reflect the light source away from the viewer receive no contribution. Under the Phong lighting model, one or more specular components are combined to determine the color and intensity of specular highlighting for each point on the object, and then are added to the result of the Lambert lighting model to produce the final color of the pixel.

For more information about the Lambert lighting model, see [How to: Create a basic Lambert shader](#).

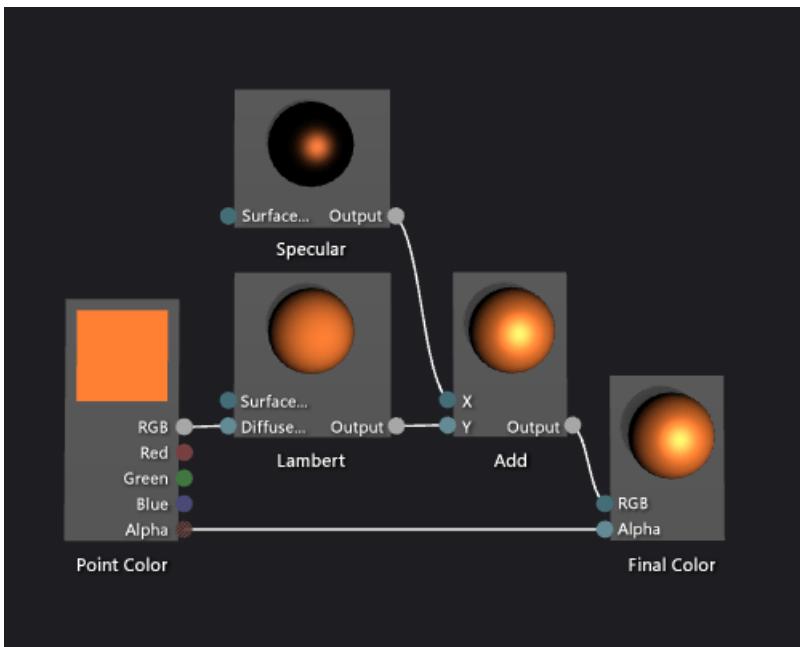
Before you begin, make sure that the **Properties** window and the **Toolbox** are displayed.

1. Create a Lambert shader, as described in [How to: Create a basic Lambert shader](#).
2. Disconnect the **Lambert** node from the **Final Color** node. Choose the **RGB** terminal of the **Lambert** node, and then choose **Break Links**. This makes room for the node that's added in the next step.
3. Add an **Add** node to the graph. In the **Toolbox**, under **Math**, select **Add** and move it to the design surface.
4. Add a **Specular** node to the graph. In the **Toolbox**, under **Utility**, select **Specular** and move it to the design surface.
5. Add the specular contribution. Move the **Output** terminal of the **Specular** node to the **X** terminal of the **Add** node, and then move the **Output** terminal of the **Lambert** node to the **Y** terminal of the **Add** node. These connections combine the total diffuse and specular color contributions for the pixel.
6. Connect the computed color value to the final color. Move the **Output** terminal of the **Add** node to the **RGB** terminal of the **Final Color** node.

The following illustration shows the completed shader graph and a preview of the shader applied to a teapot model.

NOTE

To better demonstrate the effect of the shader in this illustration, an orange color has been specified by using the **MaterialDiffuse** parameter of the shader, and a metallic-looking finish has been specified by using the **MaterialSpecular** and **MaterialSpecularPower** parameters. For information about material parameters, see the Previewing Shaders section in [Shader Designer](#).



Certain shapes might provide better previews for some shaders. For more information about how to preview shaders in the Shader Designer, see the Previewing Shaders section in [Shader Designer](#)

The following illustration shows the shader that's described in this document applied to a 3D model. The **MaterialSpecular** property is set to (1.00, 0.50, 0.20, 0.00), and its **MaterialSpecularPower** property is set to 16.

NOTE

The **MaterialSpecular** property determines the apparent finish of the surface material. A high-gloss surface such as glass or plastic tends to have a specular color that is a bright shade of white. A metallic surface tends to have a specular color that is close to its diffuse color. A satin-finish surface tends to have a specular color that is a dark shade of gray.

The **MaterialSpecularPower** property determines how intense the specular highlights are. High specular powers simulate duller, more-localized highlights. Very low specular powers simulate intense, sweeping highlights that can oversaturate and conceal the color of the whole surface.



For more information about how to apply a shader to a 3D model, see [How to: Apply a shader to a 3D model](#).

See also

- [How to: Apply a shader to a 3D model](#)
- [How to: Export a shader](#)
- [How to: Create a basic Lambert shader](#)
- [Shader Designer](#)
- [Shader Designer nodes](#)

How to: Create a basic texture shader

10/18/2019 • 2 minutes to read • [Edit Online](#)

This article demonstrates how to use the Shader Designer and the Directed Graph Shader Language (DGSL) to create a single-texture shader. This shader sets the final color directly to the RGB and alpha values that are sampled from the texture.

Create a basic texture shader

You can implement a basic, single-texture shader by writing the color and alpha values of a texture sample directly to the final output color.

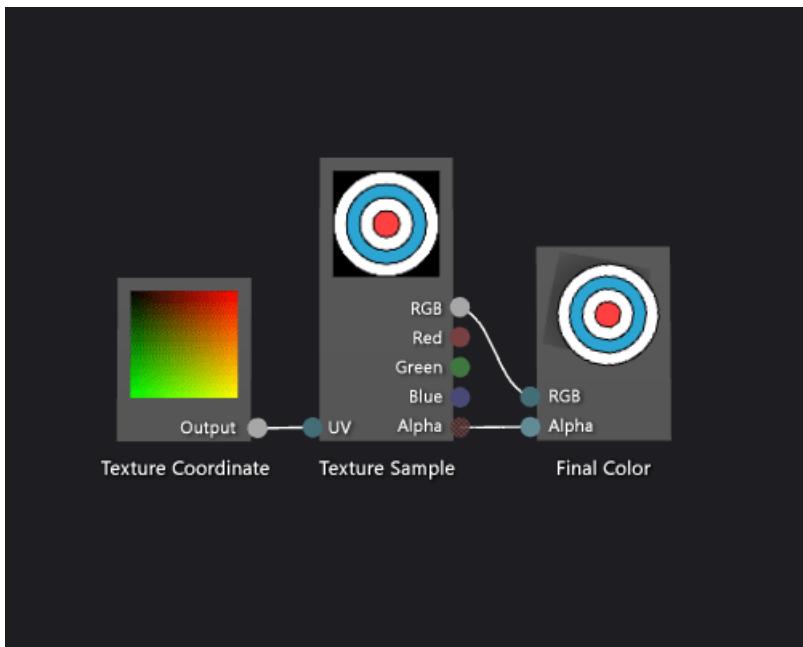
Before you begin, make sure that the **Properties** window and the **Toolbox** are displayed.

1. Create a DGSL shader to work with. For information about how to add a DGSL shader to your project, see the Getting Started section in [Shader Designer](#).
2. Delete the **Point Color** node. In **Select** mode, select the **Point Color** node, and then on the menu bar, choose **Edit** > **Delete**. This makes room for the node that's added in the next step.
3. Add a **Texture Sample** node to the graph. In the **Toolbox**, under **Texture**, select **Texture Sample** and move it to the design surface.
4. Add a **Texture Coordinate** node to the graph. In the **Toolbox**, under **Texture**, select **Texture Coordinate** and move it to the design surface.
5. Choose a texture to apply. In **Select** mode, select the **Texture Sample** node, and then in the **Properties** window, specify the texture that you want to use by using the **Filename** property.
6. Make the texture publicly accessible. Select the **Texture Sample** node, and then in the **Properties** window, set the **Access** property to **Public**. Now you can set the texture from another tool, such as the **Model Editor**.
7. Connect the texture coordinates to the texture sample. In **Select** mode, move the **Output** terminal of the **Texture Coordinate** node to the **UV** terminal of the **Texture Sample** node. This connection samples the texture at the specified coordinates.
8. Connect the texture sample to the final color. Move the **RGB** terminal of the **Texture Sample** node to the **RGB** terminal of the **Final Color** node, and then move the **Alpha** terminal of the **Texture Sample** node to the **Alpha** terminal of the **Final Color** node.

The following illustration shows the completed shader graph and a preview of the shader applied to a cube.

NOTE

In this illustration, a plane is used as the preview shape, and a texture has been specified to better demonstrate the effect of the shader.



Certain shapes might provide better previews for some shaders. For more information about how to preview shaders in the Shader Designer, see [Shader Designer](#)

See also

- [How to: Apply a shader to a 3D model](#)
- [Image Editor](#)
- [Shader Designer](#)
- [Shader Designer nodes](#)

How to: Create a grayscale texture shader

10/18/2019 • 2 minutes to read • [Edit Online](#)

This article demonstrates how to use the Shader Designer and the Directed Graph Shader Language (DGSL) to create a grayscale texture shader. This shader modifies the RGB color value of the texture sample, and then uses it together with the unmodified alpha value to set the final color.

Create a grayscale texture shader

You can implement a grayscale texture shader by modifying the color value of a texture sample before you write it to the final output color.

Before you begin, make sure that the **Properties** window and the **Toolbox** are displayed.

1. Create a basic texture shader, as described in [How to: Create a basic texture shader](#).
2. Disconnect the **RGB** terminal of the **Texture Sample** node from the **RGB** terminal of the **Final Color** node. In **Select** mode, choose the **RGB** terminal of the **Texture Sample** node, and then choose **Break Links**. This makes room for the node that's added in the next step.
3. Add a **Desaturate** node to the graph. In the **Toolbox**, under **Filters**, select **Desaturate** and move it to the design surface.
4. Calculate the grayscale value by using the **Desaturate** node. In **Select** mode, move the **RGB** terminal of the **Texture Sample** node to the **RGB** terminal of the **Desaturate** node.

NOTE

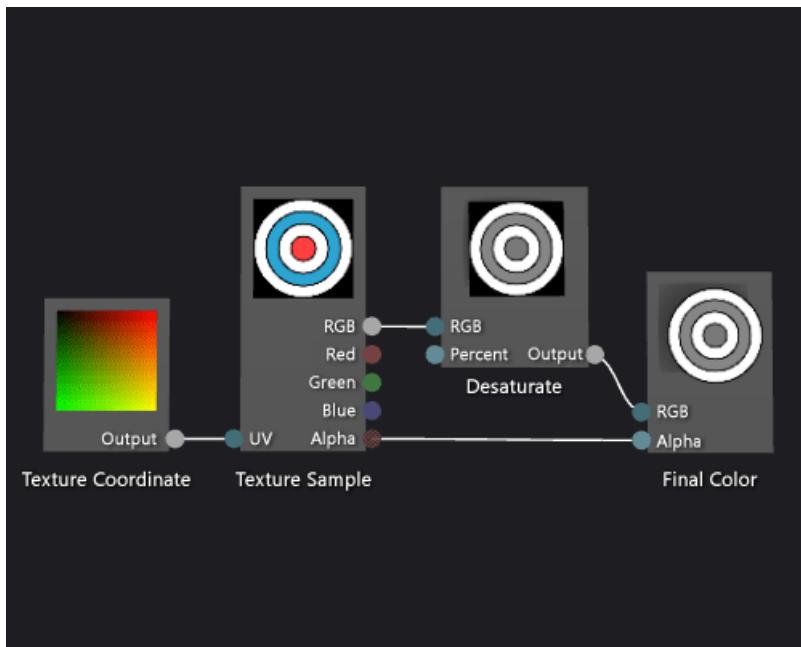
By default, the **Desaturate** node fully desaturates the input color, and uses the standard luminance weights for greyscale conversion. You can change how the **Desaturate** node behaves by changing the value of the **Luminance** property, or by only partially desaturating the input color. To partially desaturate the input color, provide a scalar value in the range [0,1] to the **Percent** terminal of the **Desaturate** node.

5. Connect the grayscale color value to the final color. Move the **Output** terminal of the **Desaturate** node to the **RGB** terminal of the **Final Color** node.

The following illustration shows the completed shader graph and a preview of the shader applied to a cube.

NOTE

In this illustration, a plane is used as the preview shape, and a texture has been specified to better demonstrate the effect of the shader.



Certain shapes might provide better previews for some shaders. For more information about previewing shaders in the Shader Designer, see [Shader Designer](#).

See also

- [How to: Apply a shader to a 3D model](#)
- [How to: Export a shader](#)
- [Image Editor](#)
- [Shader Designer](#)
- [Shader Designer nodes](#)

How to: Create a geometry-based gradient shader

10/18/2019 • 3 minutes to read • [Edit Online](#)

This article demonstrates how to use the Shader Designer and the Directed Graph Shader Language to create a geometry-based gradient shader. This shader scales a constant RGB color value by the height of each point of an object in world space.

Create a geometry-based gradient shader

You can implement a geometry-based shader by incorporating the position of the pixel into your shader. In shading languages, a pixel contains more information than just its color and location on a 2D screen. A pixel—known as a *fragment* in some systems—is a collection of values that describe the surface that corresponds to a pixel. The shader that's described in this document utilizes the height of each pixel of a 3D object in world space to affect the final output color of the fragment.

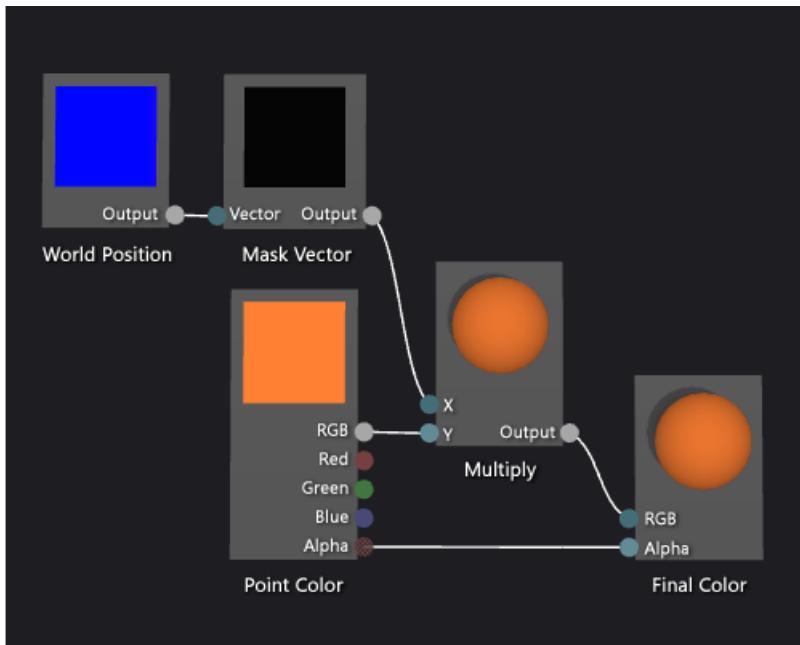
Before you begin, make sure that the **Properties** window and the **Toolbox** are displayed.

1. Create a DGSL shader with which to work. For information about how to add a DGSL shader to your project, see the Getting Started section in [Shader Designer](#).
2. Disconnect the **Point Color** node from the **Final Color** node. Choose the **RGB** terminal of the **Point Color** node, and then choose **Break Links**. This makes room for the node that's added in the next step.
3. Add a **Multiply** node to the graph. In the **Toolbox**, under **Math**, select **Multiply** and move it to the design surface.
4. Add a **Mask Vector** node to the graph. In the **Toolbox**, under **Utility**, select **Mask Vector** and move it to the design surface.
5. Specify mask values for the **Mask Vector** node. In **Select** mode, select the **Mask Vector** node, and then in the **Properties** window, set the **Green / Y** property to **True**, and then set the **Red / X**, **Blue / Z** and **Alpha / W** properties to **False**. In this example, the **Red / X**, **Green / Y**, and **Blue / Z** properties correspond to the x, y, and z components of the **World Position** node, and **Alpha / W** is unused. Because only **Green / Y** is set to **True**, only the y component of the input vector remains after it is masked.
6. Add a **World Position** node to the graph. In the **Toolbox**, under **Constants**, select **World Position** and move it to the design surface.
7. Mask the world space position of the fragment. In **Select** mode, move the **Output** terminal of the **World Position** node to the **Vector** terminal of the **Mask Vector** node. This connection masks the position of the fragment to ignore the x and z components.
8. Multiply the RGB color constant by the masked world space position. Move the **RGB** terminal of the **Point Color** node to the **Y** terminal of the **Multiply** node, and then move the **Output** terminal of the **Mask Vector** node to the **X** terminal of the **Multiply** node. This connection scales the color value by the height of the pixel in world space.
9. Connect the scaled color value to the final color. Move the **Output** terminal of the **Multiply** node to the **RGB** terminal of the **Final Color** node.

The following illustration shows the completed shader graph and a preview of the shader applied to a sphere.

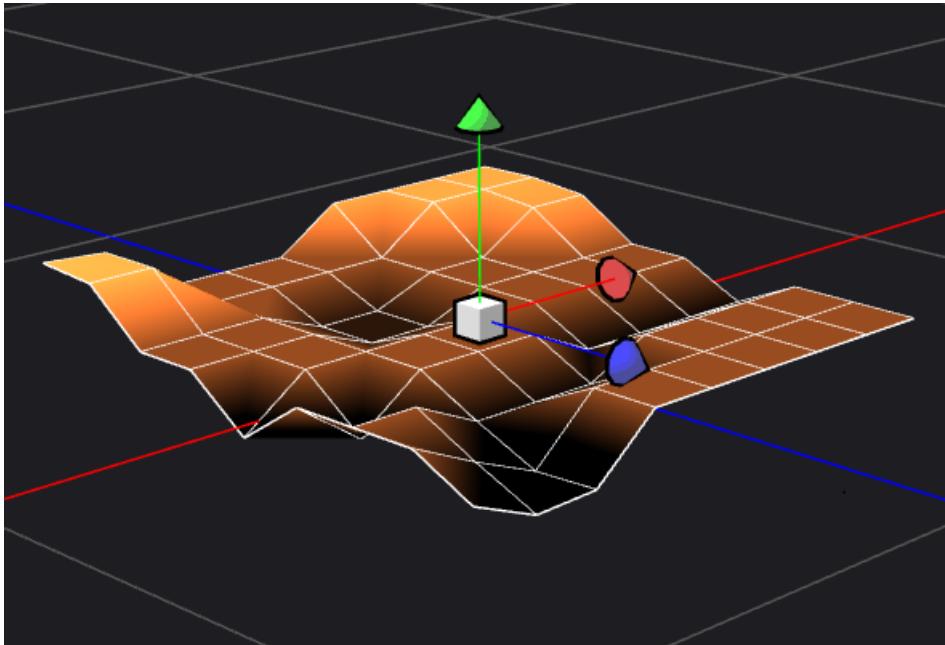
NOTE

In this illustration, an orange color is specified to better demonstrate the effect of the shader, but because the preview shape has no position in world-space, the shader cannot be fully previewed in the Shader Designer. The shader must be previewed in a real scene to demonstrate the full effect.



Certain shapes might provide better previews for some shaders. For information about how to preview shaders in the Shader Designer, see [Previewing shaders](#) in [Shader Designer](#).

The following illustration shows the shader that's described in this document applied to the 3D scene that's demonstrated in [How to: Model 3D terrain](#). The intensity of the color increases with the height of the point in the world.



For more information about how to apply a shader to a 3D model, see [How to: Apply a shader to a 3D model](#).

See also

- [How to: Apply a shader to a 3D model](#)
- [How to: Export a shader](#)

- [How to: Model 3D terrain](#)
- [How to: Create a grayscale texture shader](#)
- [Shader Designer](#)
- [Shader Designer nodes](#)

Walkthrough: Create a realistic 3D billiard ball

10/18/2019 • 13 minutes to read • [Edit Online](#)

This walkthrough demonstrates how to create a realistic 3D billiard ball by using the Shader Designer and Image Editor in Visual Studio. The 3D appearance of the billiard ball is achieved by combining several shader techniques with appropriate texture resources.

Prerequisites

You need the following components and skills to complete this walkthrough:

- A tool for assembling textures into a cube map, such as the DirectX Texture Tool that is included in the June 2010 DirectX SDK.
- Familiarity with the Image Editor in Visual Studio.
- Familiarity with the Shader Designer in Visual Studio.

Create the basic appearance with shape and texture

In computer graphics, the most-basic elements of appearance are shape and color. In a computer simulation, it is common to use a 3D model to represent the shape of a real-world object. Color detail is then applied to the surface of the model by using a texture map.

Typically, you might have to ask an artist to create a 3D model that you can use, but because a billiard ball is a common shape (a sphere), the Shader Designer already has a suitable model built in.

The sphere is the default preview shape in the Shader Designer; if you are currently using a different shape to preview your shader, switch back to the sphere.

To preview the shader by using a sphere

- On the Shader Designer toolbar, choose **Preview with sphere**.

In the next step, you'll create a shader program that applies a texture to the model, but first you have to create a texture that you can use. This walkthrough demonstrates how to create the texture by using the Image Editor, which is a part of Visual Studio, but you can use any image editor that can save the texture in a suitable format.

Make sure that the **Properties** window and the **Toolbox** are displayed.

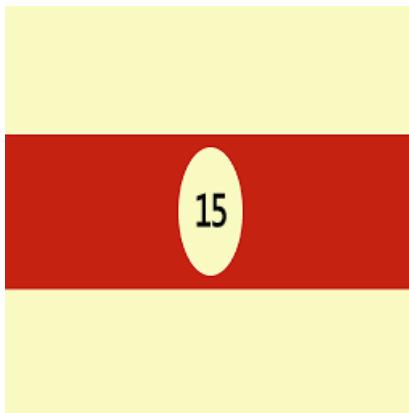
To create a billiard ball texture by using the Image Editor

1. Create a texture to work with. For information about how to add a texture to your project, see the Getting Started section in [Image Editor](#).
2. Set the image size so that its width is twice its height; this is necessary because of the way that a texture is mapped onto the spherical surface of the billiard ball. To resize the image, in the **Properties** window, specify new values for the **Width** and **Height** properties. For example, set the width to 512 and the height to 256.
3. Draw a texture for the billiard ball, keeping in mind how a texture is mapped onto a sphere.

The texture should look similar to this:

15

4. Optionally, you might want to decrease the storage requirements of this texture. You can do that by reducing the width of the texture to match its height. This compresses the texture along its width, but due to the way that the texture is mapped to the sphere, it will be expanded when the billiard ball is rendered. After resizing, the texture should look similar to this:

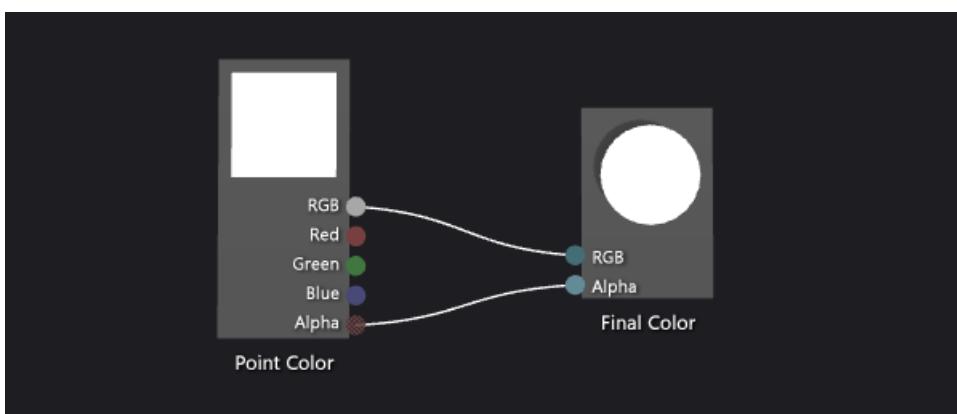


Now, you can create a shader that applies this texture to the model.

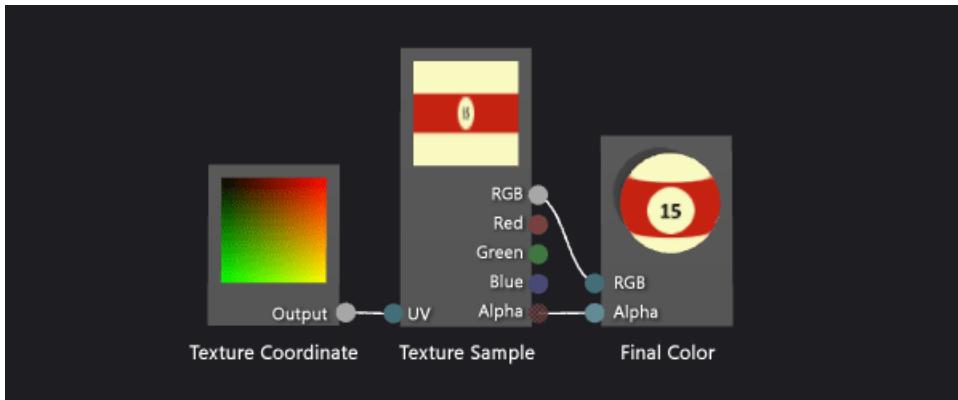
To create a basic texture shader

1. Create a DGSL shader with which to work. For information about how to add a DGSL shader to your project, see the Getting Started section in [Shader Designer](#).

By default, a shader graph looks like this:



2. Modify the default shader so that it applies the value of a texture sample to the current pixel. The shader graph should look like this:



3. Apply the texture that you created in the previous procedure by configuring the texture properties. Set the value of the **Texture** property of the **Texture Sample** node to **Texture1**, and then specify the texture file by using the **Filename** property of the **Texture1** property group in the same property window.

For more information about how to apply a texture in your shader, see [How to: Create a basic texture shader](#).

Your billiard ball should now look similar to this:



Create depth with the Lambert lighting model

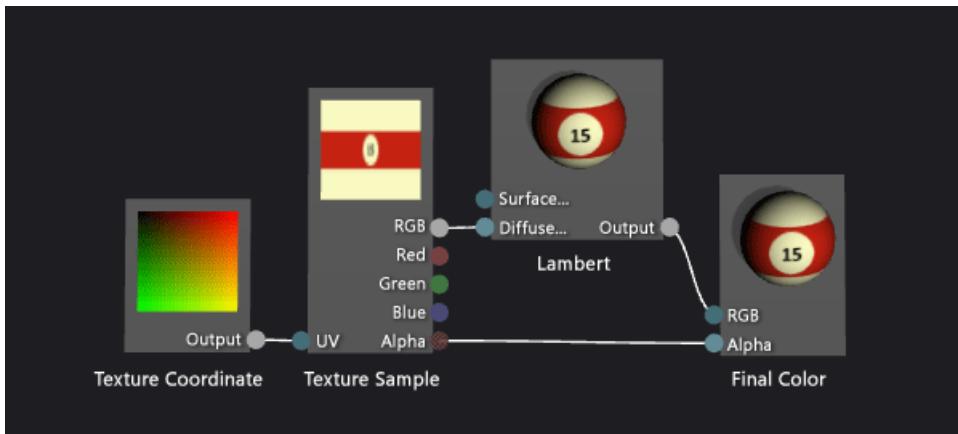
So far, you've created an easily recognizable billiard ball. However, it appears flat and uninteresting—more like a cartoon picture of a billiard ball than a convincing replica. The flat appearance results from the simplistic shader, which behaves as if each pixel on the surface of the billiard ball receives the same amount of light.

In the real world, light appears brightest on surfaces that directly face a light source and appears less bright on surfaces that are at an oblique angle to the light source. This is because the energy in the light rays is distributed across the smallest surface area when the surface directly faces the light source. As the surface turns away from the light source, the same amount of energy is distributed across an increasingly larger surface area. A surface that faces away from a light source receives no light energy at all, resulting in a completely dark appearance. This variance in brightness across the surface of an object is an important visual cue that helps indicate the shape of an object; without it, the object appears flat.

In computer graphics, *lighting models*—simplified approximations of complex, real-world lighting interactions—are used to replicate the appearance of real-world lighting. The Lambert lighting model varies the amount of diffusely reflected light across the surface of an object as described in the previous paragraph. You can add the Lambert lighting model to your shader to give the billiard ball a more convincing 3D appearance.

To add Lambert lighting to your shader

- Modify your shader to modulate the value of the texture sample by the Lambert lighting value. Your shader graph should look like this:



- Optionally, you can adjust how the lighting behaves by configuring the **MaterialDiffuse** property of the shader graph. To access properties of the shader graph, choose an empty area of the design surface, and then locate the property that you want to access in the **Properties** window.

For more information about how to apply Lambert lighting in your shader, see [How to: Create a basic Lambert shader](#).

With Lambert lighting applied, your billiard ball should look similar to this:



Enhance the basic appearance with specular highlights

The Lambert lighting model provides the sense of shape and dimension that was absent in the texture-only shader. However, the billiard ball still has a somewhat dull appearance.

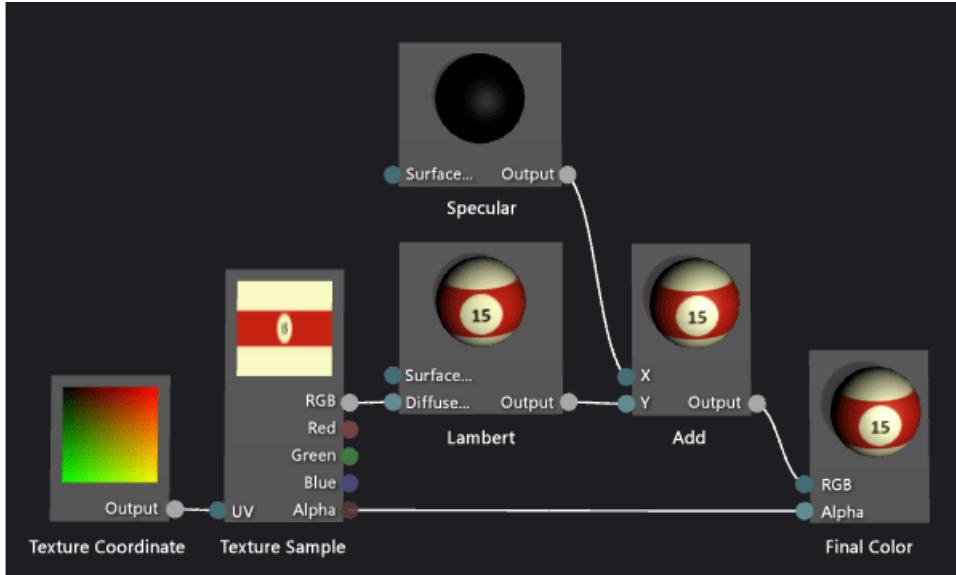
A real billiard ball usually has a glossy finish that reflects a portion of the light that falls on it. Some of this reflected light results in specular highlights, which simulate the reflecting properties of a surface. Depending on the properties of the finish, the highlights can be localized or broad, intense or subtle. These specular reflections are modeled by using the relationship between a light source, the orientation of the surface, and the camera position—that is, the highlight is most intense when the orientation of the surface reflects the light source directly into the camera, and is less intense when the reflection is less direct.

The Phong lighting model builds on the Lambert lighting model to include specular highlights as described in the

previous paragraph. You can add the Phong lighting model to your shader to give the billiard ball a simulated finish that results in a more interesting appearance.

To add specular highlights to your shader

1. Modify your shader to include the specular contribution by using additive blending. Your shader graph should look like this:



2. Optionally, you can adjust the way that the specular highlight behaves by configuring the specular properties (**MaterialSpecular** and **MaterialSpecularPower**) of the shader graph. To access properties of the shader graph, choose an empty area of the design surface, and then in the **Properties** window, locate the property that you want to access.

For more information about how to apply specular highlights in your shader, see [How to: Create a basic Phong shader](#).

With specular highlighting applied, your billiard ball should look similar to this:



Create a sense of space by reflecting the environment

With specular highlights applied, your billiard ball looks pretty convincing. It's got the right shape, the right paint job, and the right finish. However, there's still one more technique that will make your billiard ball look more like a part of its environment.

If you examine a real billiard ball closely, you can see that its glossy surface doesn't just exhibit specular highlights but also faintly reflects an image of the world around it. You can simulate this reflection by using an image of the environment as a texture and combining it with the model's own texture to determine the final color of each pixel. Depending on the kind of finish you want, you can combine more or less of the reflection texture together with the rest of the shader. For example, a shader that simulates a highly reflective surface like a mirror might use only the reflection texture, but a shader that simulates a more subtle reflection like the one found on a billiard ball might combine just a small portion of the reflection texture's value together with the rest of the shader calculation.

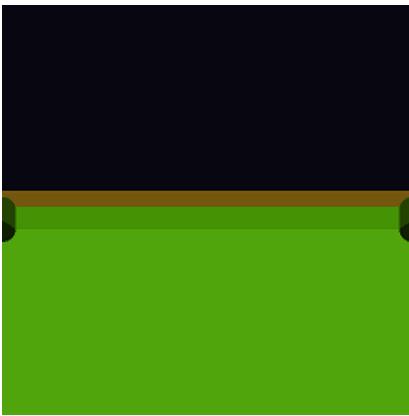
Of course, you can't just apply the reflected image to the model in the same way that you apply the model's texture map. If you did, the reflection of the world would move with the billiard ball as if the reflection were glued to it. Because a reflection can come from any direction, you need a way to provide a reflection map value for any angle, and a way to keep the reflection map oriented according to the world. To satisfy these requirements you can use a special kind of texture map—called a *cube map*—that provides six textures arranged to form the sides of a cube. From inside this cube, you can point in any direction to find a texture value. If the textures on each side of the cube contain images of the environment, you can simulate any reflection by sampling the correct location on the surface of the cube. By keeping the cube aligned to the world, you'll get an accurate reflection of the environment. To determine where the cube should be sampled, you just calculate the reflection of the camera vector off the surface of the object, and then use it as 3D texture coordinates. Using cube maps in this way is a common technique that's known as *environment mapping*.

Environment mapping provides an efficient approximation of real reflections as described in the preceding paragraphs. You can blend environment-mapped reflections into your shader to give the billiard ball a simulated finish that makes the billiard ball seem more grounded in the scene.

The first step is to create a cube map texture. In many kinds of apps, the contents of the cube map don't have to be perfect to be effective, especially when the reflection is subtle or doesn't occupy a prominent space on the screen. For example, many games use pre-computed cube maps for environment mapping and just use the one nearest to each reflective object, although this means that the reflection isn't correct. Even a rough approximation is often good enough for a convincing effect.

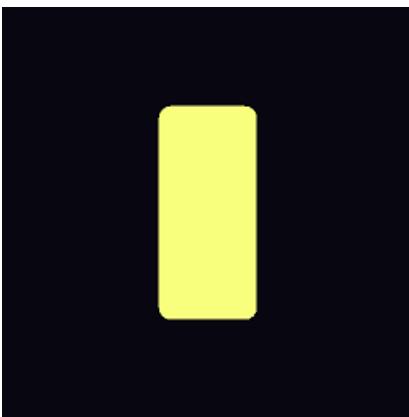
To create textures for an environment map by using the Image Editor

1. Create a texture to work with. For information about how to add a texture to your project, see the Getting Started section in [Image Editor](#).
2. Set the image size so that its width is equal to its height, and is a power of two in size; this is necessary because of the way that a cube map is indexed. To resize the image, in the **Properties** window, specify new values for the **Width** and **Height** properties. For example, set the value of the **Width** and **Height** properties to 256.
3. Use a solid color to fill the texture. This texture will be the bottom of the cube map, which corresponds to the surface of the billiard table. Keep the color you used in mind for the next texture.
4. Create a second texture that is the same size as the first. This texture will be repeated on the four sides of the cube map, which correspond to the surface and sides of a billiard table, and to the area around the billiard table. Make sure to draw the surface of the billiard table in this texture by using the same color as in the bottom texture. The texture should look similar to this:



Remember that a reflection map doesn't have to be photorealistic to be effective; for example, the cube map used to create the images in this article contains just four pockets instead of six.

5. Create a third texture that is the same size as the others. This texture will be the top of the cube map, which corresponds to the ceiling above the billiard table. To make this part of the reflection more interesting, you can draw an overhead light to reinforce the specular highlights that you added to the shader in the previous procedure. The texture should look similar to this:



Now that you have created individual textures for the sides of the cube map, you can use a tool to assemble them into a cube map that can be stored in a single .dds texture. You can use any program you want to create the cube map as long as it can save the cube map in the .dds texture format. This walkthrough demonstrates how to create the texture by using the DirectX Texture Tool that's a part of the June, 2010 DirectX SDK.

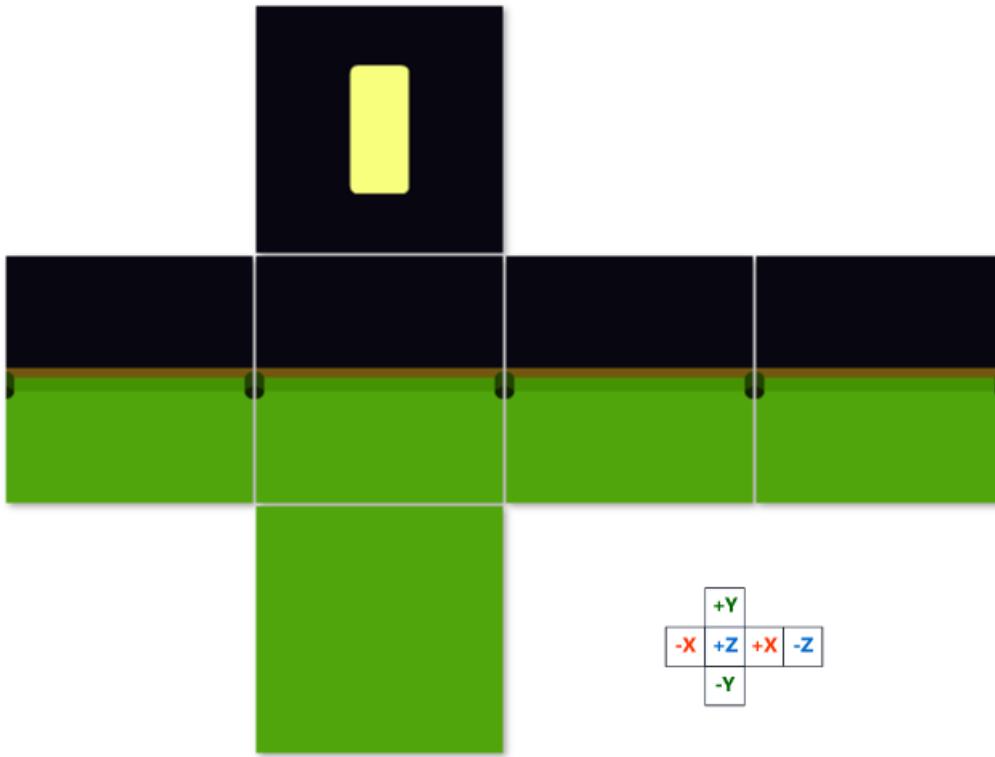
To assemble a cube map by using the DirectX Texture Tool

1. In the DirectX Texture Tool, on the main menu, choose **File > New Texture**. The **New Texture** dialog box appears.
2. In the **Texture Type** group, choose **Cubemap Texture**.
3. In the **Dimensions** group, enter the correct value for the **Width** and **Height**, and then choose **OK**. A new texture document appears. By default, the texture first shown in the texture document corresponds to the **Positive X** cube face.
4. Load the texture that you created for the side of the texture cube onto the cube face. On the main menu, choose **File > Open Onto This Cubemap Face**, select the texture that you created for the side of the cube, and then choose **Open**.
5. Repeat step 4 for the **Negative X**, **Positive Z**, and **Negative Z** cube faces. To do so, you must view the face that you want to load. To view a different cube map face, on the main menu, choose **View > Cube Map Face**, and then select the face that you want to view.
6. For the **Positive Y** cube face, load the texture that you created for the top of the texture cube.

7. For the **Negative Y** cube face, load the texture that you created for the bottom of the texture cube.

8. Save the texture.

You can imagine the layout of the cube map like this:

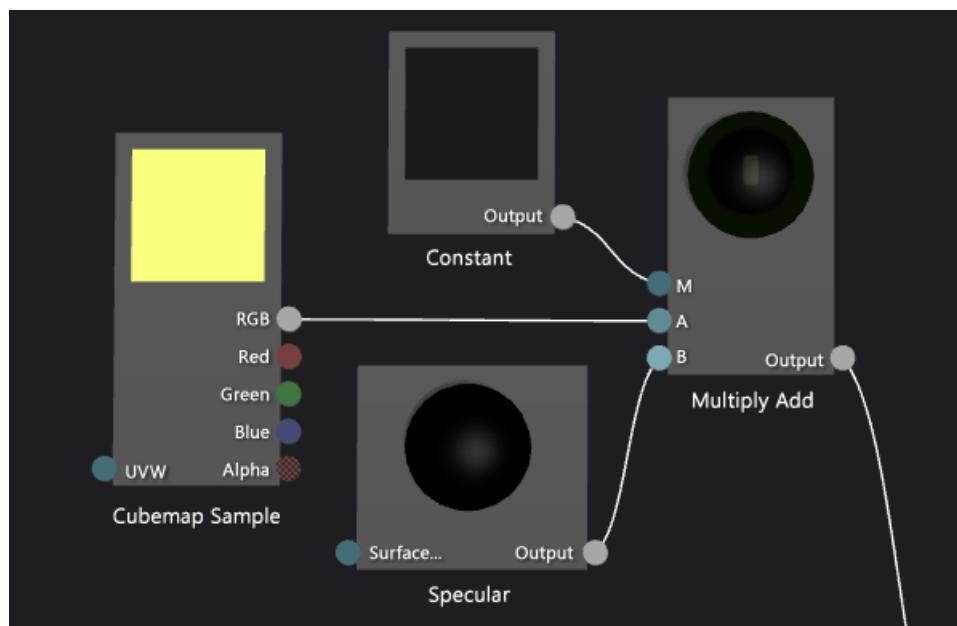


The image at the top is the positive Y (+Y) cube face; in the middle, from left to right, is the -X, +Z, +X, and -Z cube faces; at the bottom is the -Y cube face.

Now you can modify the shader to blend the cube map sample into the rest of the shader.

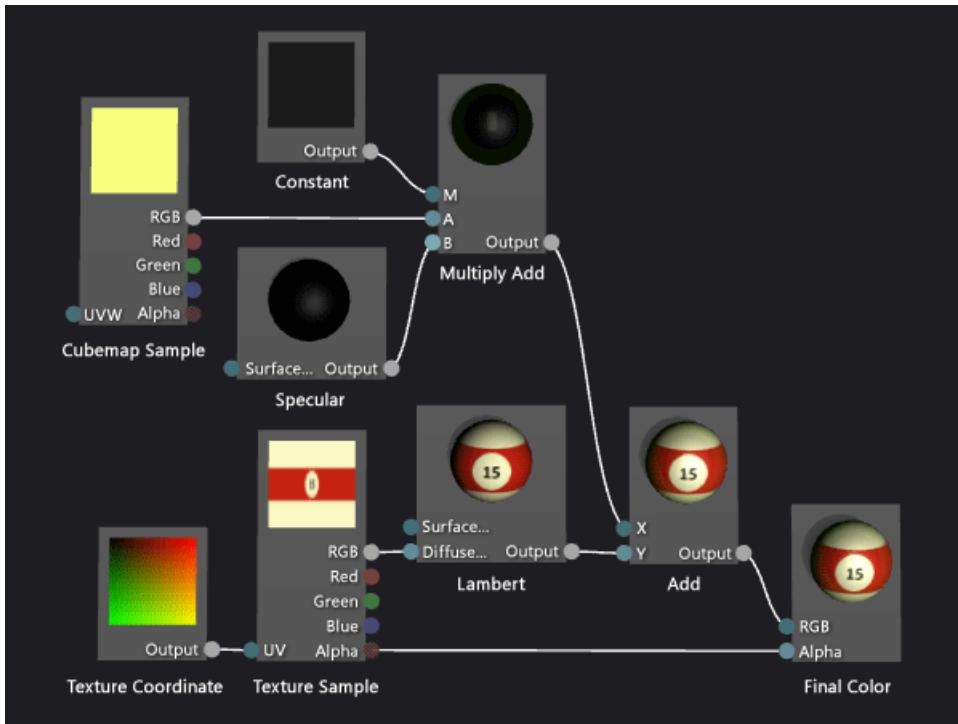
To add environment mapping to your shader

1. Modify your shader to include the environment mapping contribution by using additive blending. Your shader graph should look like this:



Note that you can use a **Multiply-Add** node to simplify the shader graph.

Here's a more detailed view of the shader nodes that implement environment mapping:



2. Apply the texture that you created in the previous procedure by configuring the texture properties of the cube map. Set the value of the **Texture** property of the **Cubemap Sample** node to **Texture2**, and then specify the texture file by using the **Filename** property of the **Texture2** property group.
3. Optionally, you can adjust the reflectivity of the billiard ball by configuring the **Output** property of the **Constant** node. To access properties of the node, select it and then in the **Properties** window, locate the property that you want to access.

With environment mapping applied, your billiard ball should look similar to this:



In this final image, notice how the effects that you added come together to create a very convincing billiard ball. The shape, texture, and lighting create the basic appearance of a 3D object, and the specular highlights and reflections make the billiard ball more interesting and look like a part of its environment.

See also

- [How to: Export a shader](#)
- [How to: Apply a shader to a 3D model](#)

- [Shader Designer](#)
- [Image Editor](#)
- [Shader Designer nodes](#)

How to: Export a shader

10/18/2019 • 2 minutes to read • [Edit Online](#)

This article demonstrates how to use the **Shader Designer** to export a Directed Graph Shader Language (DGSL) shader so that you can use it in your app.

Export a shader

After you create a shader by using the Shader Designer and before you can use it in your app, you have to export it in a format that your graphics API understands. You can export a shader in different ways to meet different needs.

1. In Visual Studio, open a **Visual Shader Graph (.dgsl)** file.

If you don't have a **Visual Shader Graph (.dgsl)** file to open, create one as described in [How to: Create a basic color shader](#).

2. On the **Shader Designer** toolbar, choose **Advanced > Export > Export As**. The **Export Shader** dialog box appears.
3. In the **Save as type** drop-down list, choose the format that you want to export.

Here are the formats that you can choose:

HLSL Pixel Shader (*.hlsl) Exports the shader as High Level Shader Language (HLSL) source code. This option makes it possible to modify the shader later, even after it's deployed in an app. This can make it easier to debug and patch the code based on end-user problems, but it also makes it easier for a user to modify your shader in unwanted ways—for example, to gain an unfair advantage in a competitive game. It also might increase the load time of the shader.

Compiled Pixel Shader (*.cso) Exports the shader as HLSL bytecode. This option makes it possible to modify the shader later, even after it's deployed in an app. This can make it easier to debug and patch the code based on end-user problems, but because the shader is pre-compiled, it does not incur extra runtime overhead when the shader is loaded by the app. Sufficiently skilled users can still modify the shader in unwanted ways, but compiling the shader makes this significantly more difficult.

C++ Header (*.h) Exports the shader as a C-style header that defines a byte array that contains HLSL bytecode. This option can make it more time-consuming to debug and patch the code based on end-user problems because the app must be recompiled to test the fix. However, because this option makes it difficult, though not impossible, to modify the shader after it's deployed in an app, it presents the most difficulty to a user who wants to modify the shader in unwanted ways.

4. In the **File name** combo box, specify a name for the exported shader, and then choose the **Save** button.

See also

- [How to: Create a basic color shader](#)
- [Shader Designer](#)

How to: Export a texture that contains mipmaps

10/18/2019 • 2 minutes to read • [Edit Online](#)

The Image Content Pipeline can generate mipmaps from a source image as part of your project's build phase. To achieve certain effects, sometimes you have to specify the image content of each MIP level manually. When you don't need to specify the image content of each MIP level manually, generating mipmaps at build time ensures that mipmap contents never become out-of-sync. It also eliminates the performance cost of generating mipmaps at run time.

This article covers:

- Configuring the source image to be processed by the Image Content Pipeline.
- Configuring the Image Content Pipeline to generate mipmaps.

Export mipmaps

Mipmapping provides automatic screen-space Level-of-Detail for textured surfaces in a 3D game or app. It enhances the rendering performance of a game or app by pre-computing down-sampled versions of a texture. Pre-computing down-sampled versions means that the entire texture does not have to be down-sampled each time it's sampled.

To export a texture that has mipmaps

1. Begin with a basic texture. Load an existing image file, or create one as described in [How to: Create a basic texture](#). To support mipmaps, specify a texture that has a width and height that are both the same power of two in size, for example, 64x64, 256x256, or 512x512.
2. Configure the texture file you just created so that it's processed by the Image Content Pipeline. In **Solution Explorer**, open the shortcut menu for the texture file you created and then choose **Properties**. On the **Configuration Properties > General** page, set the **Item Type** property to **Image Content Pipeline**. Make sure that the **Content** property is set to **Yes** and **Exclude From Build** is set to **No**. Select **Apply**.

The **Image Content Pipeline** configuration property page appears.

3. Configure the Image Content Pipeline to generate mipmaps. On the **Configuration Properties > Image Content Pipeline > General** page, set the **Generate Mips** property to **Yes (/generatemips)**.
4. Select **OK**.

When you build the project, the Image Content Pipeline converts the source image from the working format to the output format that you specified, including MIP levels. The result is copied to the project's output directory.

How to: Export a texture that has premultiplied alpha

10/18/2019 • 2 minutes to read • [Edit Online](#)

The Image Content Pipeline can generate premultiplied alpha textures from a source image. These can be simpler to use and more robust than textures that do not contain premultiplied alpha.

This document demonstrates these activities:

- Configuring the source image to be processed by the Image Content Pipeline.
- Configuring the Image Content Pipeline to generate premultiplied alpha.

Premultiplied alpha

Premultiplied alpha offers several advantages over conventional, non-premultiplied alpha, because it better represents the real-world interaction of light with physical materials by separating the texel's color contribution (the color that it adds to the scene) from its translucency (the amount of underlying color that it allows through).

Some of the advantages of using premultiplied alpha are:

- Blending with premultiplied alpha is an associative operation; the result of blending multiple translucent textures is the same, regardless of the order in which the textures are blended.
- Because of the associative nature of blending with premultiplied alpha, multi-pass rendering of translucent objects is simplified.
- By using premultiplied alpha, both pure additive blending (by setting alpha to zero) and linearly interpolated blending can be achieved simultaneously. For example, in a particle system, an additively blended fire particle can become a translucent smoke particle that's blended by using linear interpolation. Without premultiplied alpha, you would have to draw the fire particles separately from the smoke particles, and modify the render state between draw calls.
- Textures that use premultiplied alpha compress with higher quality than those that don't, and they don't exhibit the discolored edges—or "halo effect"—that can result when you blend textures that don't use premultiplied alpha.

To create a texture that uses premultiplied alpha

1. Begin with a basic texture. Load an existing image file, or create one as described in [How to: Create a basic texture](#).
2. Configure the texture file so that it's processed by the Image Content Pipeline. In **Solution Explorer**, open the shortcut menu for the texture file and then choose **Properties**. On the **Configuration Properties > General** page, set the **Item Type** property to **Image Content Pipeline**. Make sure that the **Content** property is set to **Yes** and **Exclude From Build** is set to **No**, and then choose the **Apply** button. The **Image Content Pipeline** configuration property page appears.
3. Configure the Image Content Pipeline to generate premultiplied alpha. On the **Configuration Properties > Image Content Pipeline > General** page, set the **Convert to pre-multiplied alpha format** property to **Yes (/generatepremultipliedalpha)**.
4. Choose the **OK** button.

When you build the project, the Image Content Pipeline converts the source image from the working format to the output format that you specified—this includes conversion of the image to premultiplied alpha format—and the result is copied to the project's output directory.

How to: Export a texture for use with Direct2D or JavaScript apps

10/18/2019 • 2 minutes to read • [Edit Online](#)

The Image Content Pipeline can generate textures that are compatible with Direct2D's internal rendering conventions. Textures of this kind are suitable for use in apps that use Direct2D, and in UWP apps created by using JavaScript.

This document demonstrates these activities:

- Configuring the source image to be processed by the Image Content Pipeline.
- Configuring the Image Content Pipeline to generate a texture that you can use in a Direct2D or JavaScript app.
 - Generate a block-compressed *.dds* file.
 - Generate premultiplied alpha.
 - Disable mipmap generation.

Rendering conventions in Direct2D

Textures that are used in the context of Direct2D must conform to these Direct2D internal rendering conventions:

- Direct2D implements transparency and translucency by using premultiplied alpha. Textures used with Direct2D must contain premultiplied alpha, even if the texture doesn't use transparency or translucency. For more information about premultiplied alpha, see [How to: Export a texture that has Premultiplied Alpha](#).
- The texture must be supplied in *.dds* format, by using one of these block-compression formats:
 - BC1_UNORM compression
 - BC2_UNORM compression
 - BC3_UNORM compression
- Mipmaps are not supported.

To create a texture that's compatible with Direct2D rendering conventions

1. Begin with a basic texture. Load an existing image, or create a new one as described in [How to: Create a basic texture](#). To support block-compression in *.dds* format, specify a texture that has a width and height that are multiples of four in size, for example, 100x100, 128x128, or 256x192. Because mipmapping is not supported, the texture does not have to be square and does not have to be a power of two in size.
2. Configure the texture file so that it's processed by the Image Content Pipeline. In **Solution Explorer**, open the shortcut menu for the texture file you just created and then choose **Properties**. On the **Configuration Properties > General** page, set the **Item Type** property to **Image Content Pipeline**. Make sure that the **Content** property is set to **Yes** and **Exclude From Build** is set to **No**, and then choose the **Apply** button. The **Image Content Pipeline** configuration property page appears.
3. Set the output format to one of the block-compressed formats. On the **Configuration Properties > Image Content Pipeline > General** page, set the **Compress** property to **BC3_UNORM compression (/compress:BC3_UNORM)**. You could choose any of the other BC1, BC2, or BC3 formats, depending on

your requirements. Direct2D doesn't currently support BC4, BC5, BC6, or BC7 textures. For more information about the different BC formats, see [Block compression \(Direct3D 10\)](#).

NOTE

The compression format that's specified determines the format of the file that's produced by the Image Content Pipeline. This is different than the **Format** property of the source image in the Image Editor, which determines the format of the source image file as stored on disk—that is, the *working format*. Typically, you don't want a working format that's compressed.

4. Configure the Image Content Pipeline to produce output that uses premultiplied alpha. On the **Configuration Properties > Image Content Pipeline > General** page, set the **Convert to premultiplied alpha format** property to **Yes (/generatepremultipliedalpha)**.
5. Configure the image content pipeline so that it doesn't generate mipmaps. On the **Configuration Properties > Image Content Pipeline > General** page, set the **Generate Mips** property to **No**.
6. Choose the **OK** button.

When you build the project, the Image Content Pipeline converts the source image from the working format to the output format that you specified—conversion includes generation of premultiplied alpha—and the result is copied to the project's output directory.

XML tools in Visual Studio

10/18/2019 • 2 minutes to read • [Edit Online](#)

Extensible Markup Language (XML) is a markup language that provides a format for describing data. XML separates the data and its presentation by using associated style sheets such as Extensible Stylesheet Language (XSL) and cascading style sheets (CSS). Visual Studio includes tools and features that make it easier to work with XML, XSLT, and XML schemas.

XML editor

The [XML editor](#) is used to edit XML documents. It provides full XML syntax checking, schema validation while you type, color-coding, and IntelliSense. If a schema or document type definition is provided, it is used by IntelliSense to list allowable elements and attributes.

Additional features include:

- XML snippet support, including schema-generated snippets
- Document outlining so that elements can be expanded and collapsed
- The ability to execute XSLT transformations and to view the results as text, XML, or HTML
- The ability to generate XML Schema definition language (XSD) schemas from the XML instance document
- Support for editing XSLT style sheets, including IntelliSense support
- XML Schema Explorer

XML Schema Designer

The [XML Schema Designer](#) is integrated with Visual Studio and the XML editor to enable you to work with XML schema definition language (XSD) schemas.

XSLT debugging

Visual Studio supports [debugging XSLT style sheets](#). Using the debugger, you can set break points in an XSLT style sheet, step into an XSLT style sheet from code, and so on.

NOTE

The XSLT debugger is only available in the Enterprise edition of Visual Studio.

See also

- [System.Xml](#)
- [XSLT transformations](#)
- [Process XML data using the XPath data model](#)
- [XML Document Object Model \(DOM\)](#)
- [XML Schema Object Model \(SOM\)](#)

Develop apps with the Workflow Designer

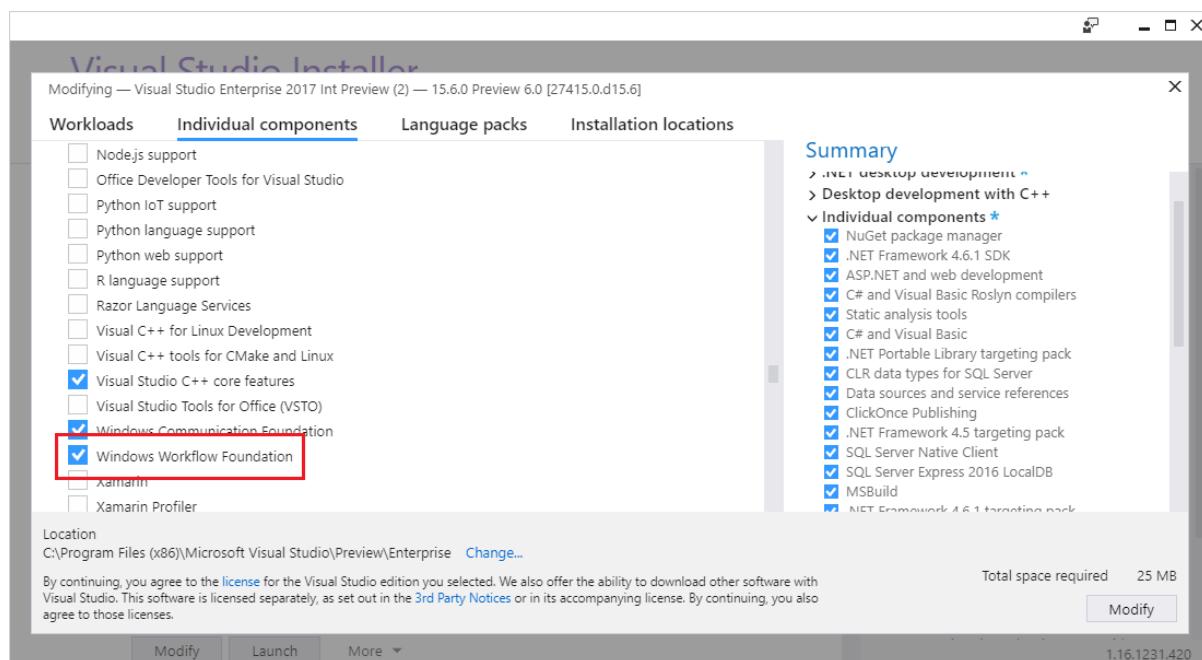
10/18/2019 • 2 minutes to read • [Edit Online](#)

The Workflow Designer is a visual designer and debugger for the graphical construction and debugging of [Windows Workflow Foundation](#) (WF) applications in Visual Studio. It enables you to compose a composite workflow application, activity library, or Windows Communication Foundation (WCF) service through the use of templates and activity designers.

Install Windows Workflow Foundation

To use Workflow project templates in Visual Studio, first install the **Windows Workflow Foundation** component.

1. Open Visual Studio Installer. A quick way to open it is by selecting **Tools > Get Tools and Features** in Visual Studio.
2. In Visual Studio Installer, select the **Individual components** tab.
3. Scroll down to the **Development activities** category and then select the **Windows Workflow Foundation** component.



4. Select **Modify**.

Visual Studio installs the **Windows Workflow Foundation** component.

See also

- [Windows Workflow Foundation \(.NET Framework\)](#)

Cross-platform mobile development in Visual Studio

10/22/2019 • 9 minutes to read • [Edit Online](#)

You can build apps for Android, iOS, and Windows devices by using Visual Studio. As you design your app, use tools in Visual Studio to easily add connected services such as Office 365, Azure App Service, and Application Insights.

Build your apps by using C# and the .NET Framework, HTML and JavaScript, or C++. Share code, strings, images, and in some cases even the user interface.

If you want to build a game or immersive graphical app, install Visual Studio tools for Unity and enjoy all of the powerful productivity features of Visual Studio with Unity, the popular cross-platform game/graphics engine and development environment for apps that run on iOS, Android, Windows, and other platforms.

Build an app for Android, iOS, and Windows (.NET Framework)



With Visual Studio Tools for Xamarin, you can target Android, iOS, and Windows in the same solution, sharing code and even UI.

LEARN MORE

[Install Visual Studio](#) (VisualStudio.com)

[Learn about Xamarin in Visual Studio](#) (VisualStudio.com)

[Xamarin mobile app development documentation](#)

[DevOps with Xamarin apps](#)

[Learn about Universal Windows apps in Visual Studio](#) (VisualStudio.com)

[Learn about the similarities between Swift and C#](#) (download.microsoft.com)

Target Android, iOS, and Windows from a single code base

You can build native apps for Android, iOS, and Windows by using C# or F# (Visual Basic is not supported at this time). To get started, install Visual Studio, select the **Mobile Development with .NET** option in the installer.

If you already have Visual Studio installed, re-run the **Visual Studio Installer** and select the same **Mobile Development with .NET** option for Xamarin (as above).

When you're done, project templates appear in the **New Project** dialog box. The easiest way to find Xamarin templates is to just search on "Xamarin."

Xamarin exposes the native functionality of Android, iOS, and Windows as .NET classes and methods. This means your apps have full access to native APIs and native controls, and they're just as responsive as apps written in the native platform languages.

After you create a project, you'll leverage all of the productivity features of Visual Studio. For example, you'll use a designer to create your pages, and use IntelliSense to explore the native API's of the mobile platforms. When

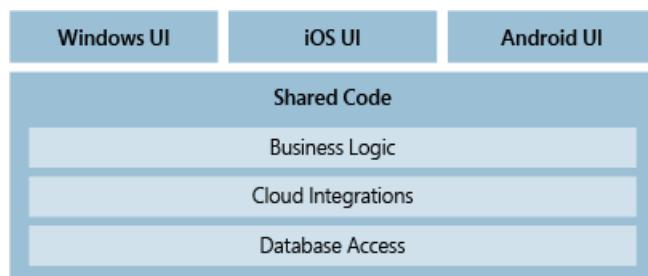
you're ready to run your app and see how it looks, you can use the Android SDK emulator and run Windows apps natively. You can also use tethered Android and Windows devices directly. For iOS projects, connect to a networked Mac and start the iOS emulator from Visual Studio, or connect to a tethered device.

Design one set of pages that render across all devices by using `Xamarin.Forms`

Depending on the complexity of your apps design, you might consider building it by using `Xamarin.Forms` templates in the **Mobile Apps** group of project templates. `Xamarin.Forms` is a UI toolkit that lets you create a single interface that you can share across Android, iOS, and Windows. When you compile a `Xamarin.Forms` solution, you'll get an Android app, an iOS app, and a Windows app. For more details, see [Learn about mobile development with Xamarin](#) and the [Xamarin.Forms documentation](#).

Share code between Android, iOS, and Windows apps

If you're not using `Xamarin.Forms` and choose to design for each platform individually, you can share most of your non-UI code between platform projects (Android, iOS, and Windows). This includes any business logic, cloud integration, database access, or any other code that targets the .NET Framework. The only code that you can't share is code that targets a specific platform.



You can share your code by using a shared project, a Portable Class Library project, or both. You might find that some code fits best in a shared project, and some code makes more sense inside a Portable Class Library project.

LEARN MORE

[Sharing Code Options \(Xamarin\)](#)

[Code sharing options with .NET](#)

Target Windows 10 devices



If you want to create a single app that targets the full breadth of Windows 10 devices, create a universal Windows app. You'll design the app by using a single project and your pages will render properly no matter what device is used to view them.

Start with a Universal Windows Platform (UWP) app project template. Design your pages visually, and then open them in a preview window to see how they appear for various types of devices. If you don't like how a page appears on a device, you can optimize the page to better fit the screen size, resolution, or various orientations such as landscape or portrait mode. You can do all of that by using intuitive tool windows and easily accessible menu options in Visual Studio. When you're ready to run your app and step through your code, you'll find all of the device emulators and simulators for different types of devices together in one drop-down list that is located on the **Standard** toolbar.

LEARN MORE

[Intro to the Universal Windows Platform](#)

[Create your first app](#)

[Develop apps for the Universal Windows Platform \(UWP\)](#)

[Migrate apps to the Universal Windows Platform \(UWP\)](#)

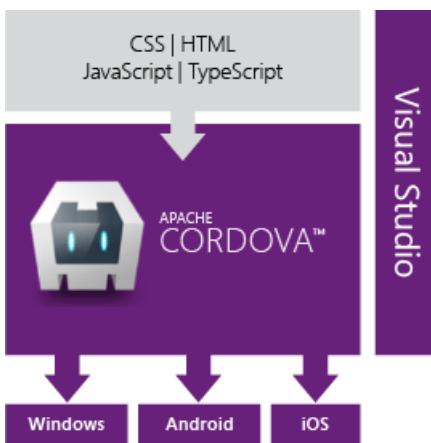
Build an app for Android, iOS, and Windows (HTML/JavaScript)



If you're a web developer, and you're familiar with HTML and JavaScript, you can target Windows, Android, and iOS by using Visual Studio Tools for Apache Cordova. These apps can target all three platforms and you can build them by using the skills and processes that you're most familiar with.

Apache Cordova is a framework that includes a plug-in model. This plug-in model provides a single JavaScript API that you can use to access the native device capabilities of all three platforms (Android, iOS, and Windows).

Because these APIs are cross-platform, you can share most of what you write between all three platforms. This reduces your development and maintenance costs. Also, there's no need to start from scratch. If you've created other types of web applications, you can share those files with your Cordova app without having to modify or redesign them in any way.



To get started, install Visual Studio and choose the **Mobile Development with Javascript** feature during setup. The Cordova tools automatically install all third-party software that's required to build your multi-platform app.

After you've installed the extension, open Visual Studio and create a **Blank App (Apache Cordova)** project. Then, you can develop your app by using JavaScript or Typescript. You can also add plug-ins to extend the functionality of your app, and APIs from plug-ins appear in IntelliSense as you write code.

When you're ready to run your app and step through your code, choose an emulator, such as the Apache Ripple emulator or Android Emulator, a browser, or a device that you've connected directly to your computer. Then, start your app. If you're developing your app on a Windows PC, you can even run it on that. All of these options are built into Visual Studio as part of the Visual Studio Tools for Apache Cordova.

Project templates for creating Universal Windows Platform (UWP) apps are still available in Visual Studio so feel free to use them if you plan to target only Windows devices. If you decide to target Android and iOS later, you can always port your code to a Cordova project.

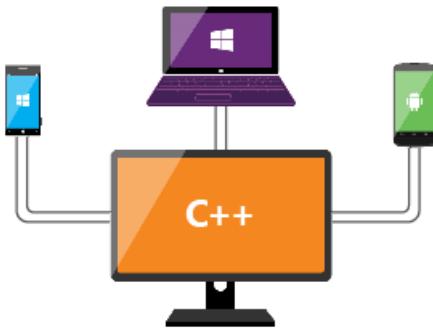
LEARN MORE

[Install Visual Studio \(VisualStudio.com\)](#)

[Get started with Visual Studio Tools for Apache Cordova](#)

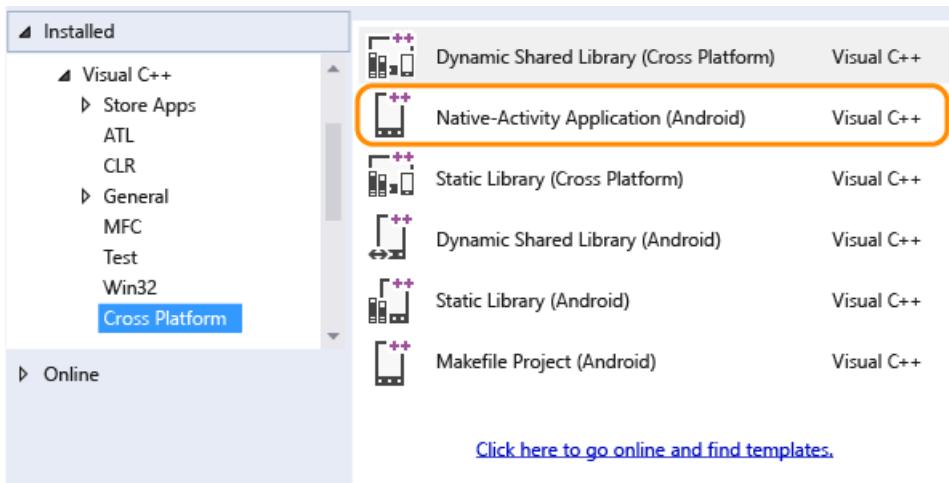
[Learn about the Visual Studio Emulator for Android \(VisualStudio.com\)](#)

Build an app for Android, iOS, and Windows (C++)



First, install Visual Studio and the **Mobile Development with C++** workload. Then, you can build a native activity application for Android, or an app that targets Windows or iOS. You can target Android, iOS, and Windows in the same solution if you want, and then share code between them by using a cross-platform static or dynamic shared library.

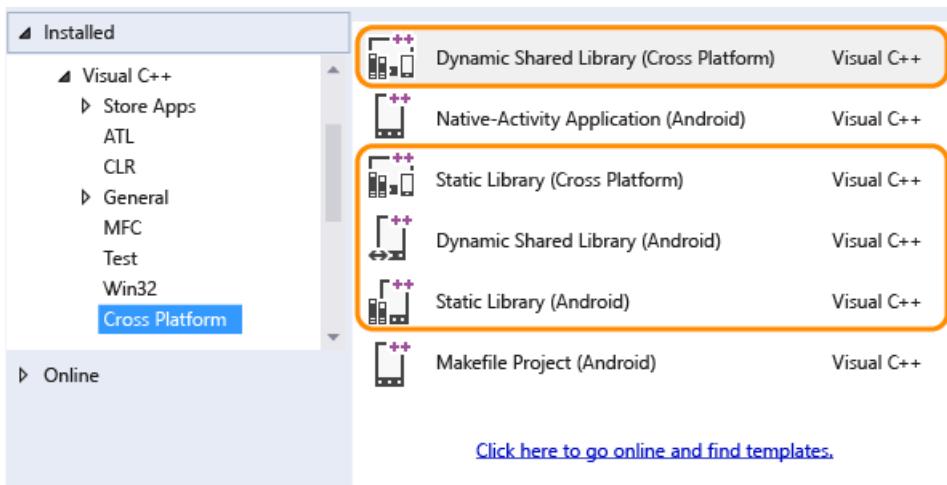
If you need to build an app for Android that requires any sort of advanced graphics manipulation, such as a game, you can use C++ to do it. Start with the **Native Activity Application (Android)** project. This project has full support for the Clang toolchain.



When you're ready to run your app and see how it looks, use the Android Emulator. It's fast, reliable, and easy to install and configure.

You can also build an app that targets the full breadth of Windows 10 devices by using C++ and a Universal Windows Platform (UWP) app project template. Read more about this in the [Target Windows 10 devices](#) section that appears earlier in this topic.

You can share C++ code between Android, iOS, and Windows by creating a static or dynamic shared library.



You can consume that library in a Windows, iOS, or Android project, like the ones described earlier in this section. You can also consume it in an app that you build by using Xamarin, Java, or any language that lets you invoke functions in an unmanaged DLL.

As you write code in these libraries, you can use IntelliSense to explore the native APIs of the Android and Windows platforms. These library projects are fully integrated with the Visual Studio debugger so you can set breakpoints, step through code, and find and fix issues by using all of the advanced features of the debugger.

LEARN MORE

[Download Visual Studio](#) (VisualStudio.com)

[Install cross-platform mobile development with C++](#)

[Learn more about using C++ to target multiple platforms](#) (VisualStudio.com)

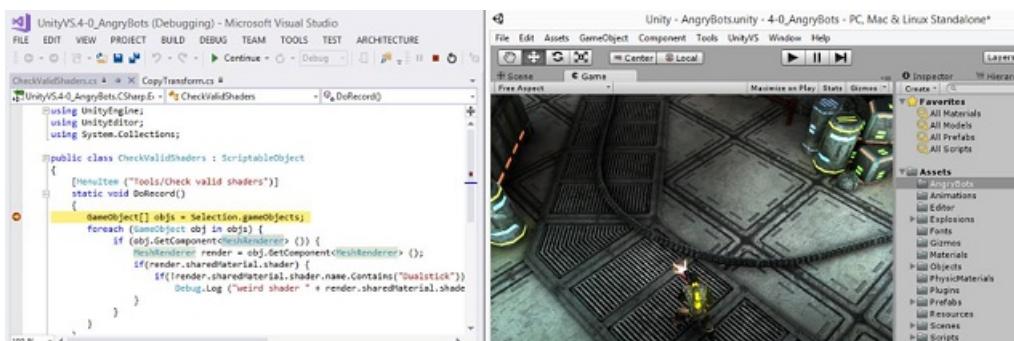
[Install what you need, and then create a native activity application for Android](#)

[Learn more about sharing C++ code with Android and Windows apps](#) (VisualStudio.com)

[Cross-platform mobile development examples for C++](#)

Build a cross-platform game for Android, iOS, and Windows by using Visual Studio tools for Unity

Visual Studio Tools for Unity is a free extension for Visual Studio that integrates Visual Studio's powerful code editing, productivity, and debugging tools with *Unity*, the popular cross-platform gaming/graphics engine and development environment for immersive apps that target Windows, iOS, Android, and other platforms including the web.



With Visual Studio Tools for Unity (VSTU), you can use Visual Studio to write game and editor scripts in C# and

then use its powerful debugger to find and fix errors. The latest release of VSTU brings support for Unity 2018.1 and includes syntax coloring for Unity's ShaderLab shader language, better synchronization with Unity, richer debugging, and improved code generation for the MonoBehavior wizard. VSTU also brings your Unity project files, console messages, and the ability to start your game into Visual Studio so you can spend less time switching to and from the Unity Editor while writing code.

LEARN MORE

[Learn more about building Unity games with Visual Studio](#)

[Read more about Visual Studio Tools for Unity](#)

[Start using Visual Studio Tools for Unity](#)

[Read about the latest enhancements to the Visual Studio Tools for Unity 2.0 Preview](#) (Visual Studio blog)

[Watch a video introduction to the Visual Studio Tools for Unity 2.0 Preview](#) (Video)

[Learn about Unity](#) (Unity website)

See also

- [Add Office 365 APIs to a Visual Studio project](#)
- [Azure App Services - Mobile Apps](#)
- [Visual Studio App Center](#)

Office and SharePoint development in Visual Studio

4/18/2019 • 3 minutes to read • [Edit Online](#)

You can extend Microsoft Office and SharePoint by creating a lightweight app or add-in that users download from the [Office Store](#) or an organizational catalog, or by creating a .NET Framework-based solution that users install on a computer.

In this topic:

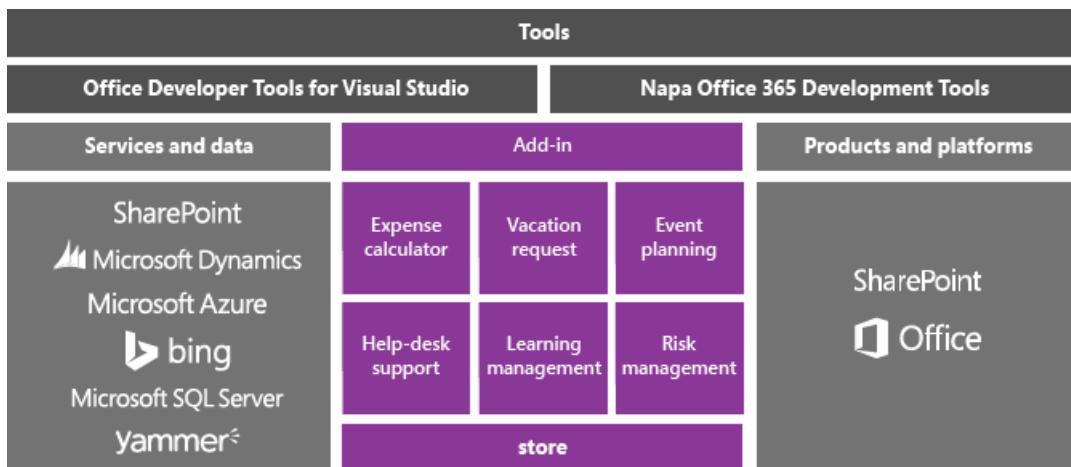
- [Create add-ins for Office and SharePoint](#)
- [Create a VSTO Add-in](#)
- [Create a SharePoint solution](#)

Create add-ins for Office and SharePoint

Office 2013 and SharePoint 2013 introduce a new add-in model that helps you build, distribute, and monetize add-ins that extend Office and SharePoint. These add-ins can run in Office or SharePoint Online, and users can interact with them from many devices.

Find out how to use the new [Office Add-in model](#) to extend the Office experience for your users.

These add-ins have small footprints compared to VSTO add-ins and solutions, and you can build them by using almost any web programming technology such as HTML5, JavaScript, CSS3, and XML. To get started, use the Office Developer Tools in Visual Studio, or the lightweight web-based tool code-named Napa Office 365 Development Tools, which lets you create projects, write code, and run your add-ins in a browser.



Build an Office Add-in

To extend the functionality of Office, build an Office add-in. It's basically a webpage that's hosted in an Office application such as Excel, Word, Outlook, and PowerPoint. Your app can add functionality to documents, worksheets, email messages, appointments, presentations, and projects.

You can sell your app in the Office Store. The [Office Store](#) makes it easy to monetize your add-ins, manage updates, and track telemetry. You can also publish your app to users through an app catalog in SharePoint, or on Exchange Server.

The following app for Office shows worksheet data in a Bing map.

A	B	C	D	E	F	G	H	I
1								
2	State	Sales						
3	California	\$800,000						
4	Pennsylvania	\$600,000						
5	Texas	\$500,000						
6	New York	\$500,000						
7								
8								
9								
10								
11								
12								
13								
14								
15								
16								
17								



Learn more

TO	SEE
Learn more about Office add-ins, and then build one.	Office add-ins
Compare the different ways in which you can extend Office, and decide whether you should use an app or an Office add-in.	Roadmap for Office Add-ins, VSTO, and VBA

Build a SharePoint Add-in

To extend SharePoint for your users, build a SharePoint add-in. It's basically a small, easy-to-use, stand-alone application that solves a need for your users or business.

You can sell your app for SharePoint in the [Office Store](#). You can also publish your add-in to users through an add-in catalog in SharePoint. Site owners can install, upgrade, and uninstall your add-in on their SharePoint sites without the help of a farm server or site collection administrator.

Here's an example of an app for SharePoint that helps users manage business contacts.

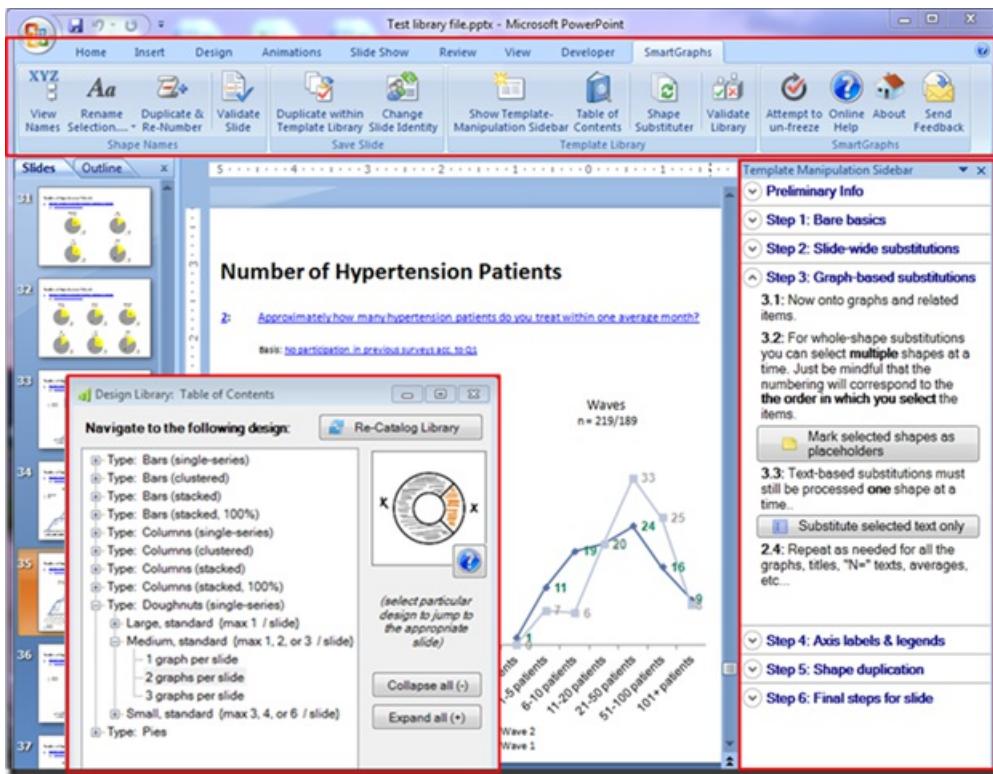
Learn more

TO	SEE
Learn more about SharePoint add-ins, and then build one.	SharePoint Add-ins
Compare add-ins for SharePoint with traditional SharePoint solutions.	SharePoint Add-ins compared with SharePoint solutions
Choose whether to build a SharePoint add-in or a SharePoint solution.	Decide between SharePoint Add-ins and SharePoint solutions

Create a VSTO Add-in

Create a VSTO add-in to target Office 2007 or Office 2010, or to extend Office 2013 and Office 2016 beyond what's possible with Office add-ins. VSTO add-ins run only on the desktop. Users have to install VSTO add-ins, so they're typically more difficult to deploy and support. However, your VSTO add-in can be integrated more closely with Office. For example, it can add tabs and controls to the Office Ribbon and perform advanced automation tasks such as merging documents or modifying charts. You can leverage the .NET Framework and use C# and Visual Basic to interact with Office objects.

Here's an example what a VSTO add-in can do. This VSTO add-in adds Ribbon controls, a custom task pane, and a dialog box to PowerPoint.



Learn more

TO	READ
Compare the different ways in which you can extend Office, and decide whether you should use a VSTO add-in or an Office add-in.	Roadmap for Office Add-ins, VSTO, and VBA
Create a VSTO add-in.	VSTO add-ins build with Visual Studio

Create a SharePoint solution

Create a SharePoint solution to target SharePoint Foundation 2010 and SharePoint Server 2010, or to extend SharePoint 2013 and SharePoint 2016 in ways beyond what's possible with a SharePoint add-in.

SharePoint solutions require on-premises SharePoint farm servers. Administrators must install them, and because solutions execute in SharePoint, they can affect the performance of the server. However, solutions provide deeper access to SharePoint objects. Also, when you build a SharePoint solution, you can leverage the .NET Framework and use C# and Visual Basic to interact with SharePoint objects.

Learn more

TO	SEE
Compare SharePoint solutions with SharePoint add-ins.	SharePoint Add-ins compared with SharePoint solutions
Create a SharePoint solution.	Create SharePoint solutions

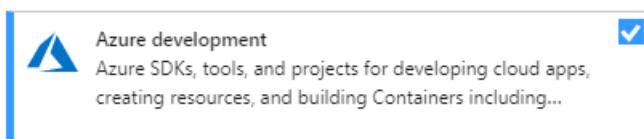
Access data in Visual Studio

10/23/2019 • 7 minutes to read • [Edit Online](#)

In Visual Studio, you can create applications that connect to data in virtually any database product or service, in any format, anywhere—on a local machine, on a local area network, or in a public, private, or hybrid cloud.

For applications in JavaScript, Python, PHP, Ruby, or C++, you connect to data like you do anything else, by obtaining libraries and writing code. For .NET applications, Visual Studio provides tools that you can use to explore data sources, create object models to store and manipulate data in memory, and bind data to the user interface. Microsoft Azure provides SDKs for .NET, Java, Node.js, PHP, Python, Ruby, and mobile apps, and tools in Visual Studio for connecting to Azure Storage.

The following lists show just a few of the many database and storage systems that can be used from Visual Studio. The [Microsoft Azure](#) offerings are data services that include all provisioning and administration of the underlying data store. The **Azure development** workload in [Visual Studio 2017](#) enables you to work with Azure data stores directly from Visual Studio.



Most of the other SQL and NoSQL database products that are listed here can be hosted on a local machine, on a local network, or in Microsoft Azure on a virtual machine. If you host the database in a Microsoft Azure virtual machine, you're responsible for managing the database itself.

Microsoft Azure

- SQL Database
- Azure Cosmos DB
- Storage (blobs, tables, queues, files)
- SQL Data Warehouse
- SQL Server Stretch Database
- StorSimple
- And more...

SQL

- SQL Server 2005-2016 (includes Express and LocalDB)
- Firebird
- MariaDB
- MySQL
- Oracle
- PostgreSQL
- SQLite
- And more...

NoSQL

- Apache Cassandra
- CouchDB

- MongoDB
- NDatabase
- OrientDB|
- RavenDB
- VelocityDB
- And more...

Many database vendors and third parties support Visual Studio integration by NuGet packages. You can explore the offerings on nuget.org or through the NuGet Package Manager in Visual Studio (**Tools > NuGet Package Manager > Manage NuGet Packages for Solution**). Other database products integrate with Visual Studio as an extension. You can browse these offerings in the [Visual Studio Marketplace](#) or by navigating to **Tools > Extensions and Updates** and then selecting **Online** in the left pane of the dialog box. For more information, see [Compatible database systems for Visual Studio](#).

Many database vendors and third parties support Visual Studio integration by NuGet packages. You can explore the offerings on nuget.org or through the NuGet Package Manager in Visual Studio (**Tools > NuGet Package Manager > Manage NuGet Packages for Solution**). Other database products integrate with Visual Studio as an extension. You can browse these offerings in the [Visual Studio Marketplace](#) or by navigating to **Extensions > Manage Extensions** and then selecting **Online** in the left pane of the dialog box. For more information, see [Compatible database systems for Visual Studio](#).

NOTE

Extended support for SQL Server 2005 ended on April 12, 2016. There is no guarantee that data tools in Visual Studio 2015 and later will continue to work with SQL Server 2005. For more information, see the [end-of-support announcement for SQL Server 2005](#).

.NET languages

All .NET data access, including in .NET Core, is based on ADO.NET, a set of classes that defines an interface for accessing any kind of data source, both relational and non-relational. Visual Studio has several tools and designers that work with ADO.NET to help you connect to databases, manipulate the data, and present the data to the user. The documentation in this section describes how to use those tools. You can also program directly against the ADO.NET command objects. For more information about calling the ADO.NET APIs directly, see [ADO.NET](#).

For data-access documentation related to ASP.NET, see [Working with Data](#) on the ASP.NET site. For a tutorial on using Entity Framework with ASP.NET MVC, see [Getting Started with Entity Framework 6 Code First using MVC 5](#).

Universal Windows Platform (UWP) apps in C# or Visual Basic can use the Microsoft Azure SDK for .NET to access Azure Storage and other Azure services. The Windows.Web.HttpClient class enables communication with any RESTful service. For more information, see [How to connect to an HTTP server using Windows.Web.Http](#).

For data storage on the local machine, the recommended approach is to use SQLite, which runs in the same process as the app. If an object-relational mapping (ORM) layer is required, you can use Entity Framework. For more information, see [Data access](#) in the Windows Developer Center.

If you are connecting to Azure services, be sure to download the latest [Azure SDK tools](#).

Data providers

For a database to be consumable in ADO.NET, it must have a custom *ADO.NET data provider* or else must expose an ODBC or OLE DB interface. Microsoft provides a [list of ADO.NET data providers](#) for SQL Server products, as well as ODBC and OLE DB providers.

Data modeling

In .NET, you have three choices for modeling and manipulating data in memory after you have retrieved it from a data source:

Entity Framework The preferred Microsoft ORM technology. You can use it to program against relational data as first-class .NET objects. For new applications, it should be the default first choice when a model is required. It requires custom support from the underlying ADO.NET provider.

LINQ to SQL An earlier-generation object-relational mapper. It works well for less complex scenarios but is no longer in active development.

Datasets The oldest of the three modeling technologies. It is designed primarily for rapid development of "forms over data" applications in which you are not processing huge amounts of data or performing complex queries or transformations. A DataSet object consists of DataTable and DataRow objects that logically resemble SQL database objects much more than .NET objects. For relatively simple applications based on SQL data sources, datasets might still be a good choice.

There is no requirement to use any of these technologies. In some scenarios, especially where performance is critical, you can simply use a DataReader object to read from the database and copy the values that you need into a collection object such as List<T>.

Native C++

C++ applications that connect to SQL Server should use the [Microsoft® ODBC Driver 13.1 for SQL Server](#) in most cases. If the servers are linked, then OLE DB is necessary and for that you use the [SQL Server Native Client](#). You can access other databases by using [ODBC](#) or OLE DB drivers directly. ODBC is the current standard database interface, but most database systems provide custom functionality that can't be accessed through the ODBC interface. OLE DB is a legacy COM data-access technology that is still supported but not recommended for new applications. For more information, see [Data Access in Visual C++](#).

C++ programs that consume REST services can use the [C++ REST SDK](#).

C++ programs that work with Microsoft Azure Storage can use the [Microsoft Azure Storage Client](#).

Data modeling—Visual Studio does not provide an ORM layer for C++. [ODB](#) is a popular open-source ORM for C++.

To learn more about connecting to databases from C++ apps, see [Visual Studio data tools for C++](#). For more information about legacy Visual C++ data-access technologies, see [Data Access](#).

JavaScript

[JavaScript in Visual Studio](#) is a first-class language for building cross-platform apps, UWP apps, cloud services, websites, and web apps. You can use Bower, Grunt, Gulp, npm, and NuGet from within Visual Studio to install your favorite JavaScript libraries and database products. Connect to Azure storage and services by downloading SDKs from the [Azure website](#). Edge.js is a library that connects server-side JavaScript (Node.js) to ADO.NET data sources.

Python

Install [Python support in Visual Studio](#) to create Python applications. The Azure documentation has several tutorials on connecting to data, including the following:

- [Django and SQL Database on Azure](#)
- [Django and MySQL on Azure](#)
- Work with [blobs, files, queues](#), and [tables \(Cosmo DB\)](#).

Related topics

[Microsoft AI platform](#)—Provides an introduction to the Microsoft intelligent cloud, including Cortana Analytics Suite and support for Internet of Things.

[Microsoft Azure Storage](#)—Describes Azure Storage, and how to create applications by using Azure blobs, tables, queues, and files.

[Azure SQL Database](#)—Describes how to connect to Azure SQL Database, a relational database as a service.

[SQL Server Data Tools](#)—Describes the tools that simplify design, exploration, testing, and deploying of data-connected applications and databases.

[ADO.NET](#)—Describes the ADO.NET architecture and how to use the ADO.NET classes to manage application data and interact with data sources and XML.

[ADO.NET Entity Framework](#)—Describes how to create data applications that allow developers to program against a conceptual model instead of directly against a relational database.

[WCF Data Services 4.5](#)—Describes how to use WCF Data Services to deploy data services on the web or an intranet that implement the [Open Data Protocol \(OData\)](#).

[Data in Office Solutions](#)—Contains links to topics that explain how data works in Office solutions. This includes information about schema-oriented programming, data caching, and server-side data access.

[LINQ \(Language-Integrated Query\)](#)—Describes the query capabilities built into C# and Visual Basic, and the common model for querying relational databases, XML documents, datasets, and in-memory collections.

[XML Tools in Visual Studio](#)—Discusses working with XML data, debugging XSLT, .NET XML features, and the architecture of XML Query.

[XML Documents and Data](#)—Provides an overview to a comprehensive and integrated set of classes that work with XML documents and data in .NET.

Resources for designing accessible applications

8/28/2019 • 2 minutes to read • [Edit Online](#)

Learn more about technologies that support accessible design. We've also included tips and links to tutorials that can help you develop accessible Windows apps and websites.

NOTE

For more information about how we develop products that empower everyone, see [Microsoft Accessibility](#).

Technologies

- **Microsoft Active Accessibility** A COM-based technology that improves the way accessibility aids work with applications running on Microsoft Windows. It provides dynamic-link libraries that are incorporated into the operating system and a COM interface. It also has application programming elements that provide methods for exposing information about user interface elements. For more information, see [Microsoft active accessibility](#).
- **Microsoft .NET Speech Technologies** The Microsoft .NET Speech SDK provides a set of Microsoft ASP.NET controls, a Microsoft Internet Explorer Speech add-in, sample applications, and documentation. Web developers can use these tools to create, debug, and deploy speech-enabled ASP.NET applications. The tools are integrated seamlessly into Microsoft Visual Studio, allowing developers to work in the familiar development environment. For more information, see [Speech server](#).
- **Understanding SAMI 1.0** Microsoft Synchronized Accessible Media Interchange (SAMI) technology provides a way for developers to caption audio content for PC multimedia. For more information, see [Understanding SAMI 1.0](#).

Windows applications

- **Walkthrough: Creating an accessible Windows-based application** This article provides step-by-step instructions for including the five accessibility requirements for the "Certified for Windows" logo in a sample Windows application.
- **Guidelines for keyboard user interface design** This technical article describes how to design a Windows application that users can navigate from the keyboard. For more information, see [Guidelines for keyboard user interface design](#).
- **Console accessibility** This technical article describes the APIs and events used to expose the console in Windows XP for accessibility aids. For more information, see [Console accessibility](#).

Websites

- **Walkthrough: Accessibility Guidelines for Using Image Controls, Menu Controls, and AutoPostBack** This article provides step-by-step instructions for including accessible controls in a sample web page. It also gives some accessibility design tips for the Web.
- **Creating Accessible Web Pages with DHTML** This technical article lists HTML 4.0 elements that are accessible as well as accessible web design tips. For more information, see [Create accessible web pages with DHTML](#).

Third-party resources

- **Web Accessibility Initiative of the World Wide Web Consortium (W3C)** This website provides guidelines and techniques for accessible website development. For more information, see <https://www.w3.org/WAI/GL/>.

See also

- [Accessibility features of Visual Studio](#)
- [Accessibility for Visual Studio for Mac](#)

Develop globalized and localized apps

10/18/2019 • 2 minutes to read • [Edit Online](#)

Visual Studio makes developing for an international audience easy by taking advantage of services built into .NET.

For example, the project system for Windows Forms apps can generate resource files for both the fallback UI culture and each additional UI culture. When you build a project in Visual Studio, the resource files are compiled from the Visual Studio XML format (.resx) to an intermediate binary format (.resources), which are then embedded in satellite assemblies. For more information, see [Resource files in Visual Studio](#) and [Create satellite assemblies for desktop apps](#).

Bidirectional languages

You can use Visual Studio to create applications that correctly display text in languages written right-to-left, including Arabic and Hebrew. For some features, you can simply set properties. In other cases, you must implement features in code.

NOTE

In order to enter and display bidirectional languages, you must be working with a version of Windows that is configured with the appropriate language. This can either be an English version of Windows with the appropriate language pack installed, or the appropriately localized version of Windows.

Apps that support bidirectional languages

- Windows apps

You can create fully bidirectional applications that include support for bidirectional text, right-to-left reading order, and mirroring (reversing the layout of windows, menus, dialog boxes, and so on). Except for mirroring, these features are available by default or as property settings. Mirroring is supported inherently for some features, such as message boxes. However, in other cases you must implement mirroring in code. For more information, see [bidirectional support for Windows Forms applications](#).

- Web apps

Web services support sending and receiving UTF-8 and Unicode text, making them suitable for applications that involve bidirectional languages. Web client applications rely on browsers for their user interface, so the degree of bidirectional support in a web application is dependent on how well the user's browser supports those bidirectional features. In Visual Studio, you can create applications with support for Arabic or Hebrew text, right-to-left reading order, file encoding, and local culture settings. For more information, see [Bidirectional support for ASP.NET web applications](#).

NOTE

Console apps do not include text support for bidirectional languages. This is a consequence of how Windows works with console applications.

See also

- [Support for bidirectional languages in Visual Studio](#)
- [Globalize and localize .NET apps](#)

- Resources in .NET apps

Compile and build in Visual Studio

9/23/2019 • 2 minutes to read • [Edit Online](#)

When you build source code, the build engine creates assemblies and executable applications. 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 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 detect and correct run-time errors, such as logic errors and semantic errors, by building and running debug versions of the code.

A successful build validates that the application's source code contains correct syntax and that all static references to libraries, assemblies, and other components can resolve. An application executable is produced that can be tested for proper functioning in both a [debugging environment](#) and through a variety of manual and automated tests to [validate code quality](#). Once the application has been fully tested, you can compile a release version to deploy to your customers. For an introduction to this process, see [Walkthrough: Building an application](#).

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	<ul style="list-style-type: none">- 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	<ul style="list-style-type: none">- Build projects without installing Visual Studio.- Run multi-processor builds for all project types.- Customize most areas of the build system.
Azure Pipelines	<ul style="list-style-type: none">- 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](#) and [Azure Pipelines](#), respectively.

NOTE

This topic applies to Visual Studio on Windows. For Visual Studio for Mac, see [Compile and build in Visual Studio for Mac](#).

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](#).

Next, see [Building and cleaning projects and solutions in Visual Studio](#) to learn about the different aspects customizations you can make to the process. Customizations include [changing output directories](#), [specifying custom build events](#), [managing project dependencies](#), [managing build log files](#), and [suppressing compiler warnings](#).

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

- [Understand build configurations](#)
- [Understand build platforms](#)
- [Manage project and solution properties.](#)
- [Specify build events in C# and Visual Basic.](#)
- [Set build options](#)
- [Build multiple projects in parallel.](#)

See also

- [Building \(compiling\) website projects](#)
- [Compile and build \(Visual Studio for Mac\)](#)

Walkthrough: Build an application

10/18/2019 • 5 minutes to read • [Edit Online](#)

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](#) sample. Choose either C# or Visual Basic. After the .zip file has downloaded, extract it and open the *ExpenseIntro.sln* file using Visual Studio.

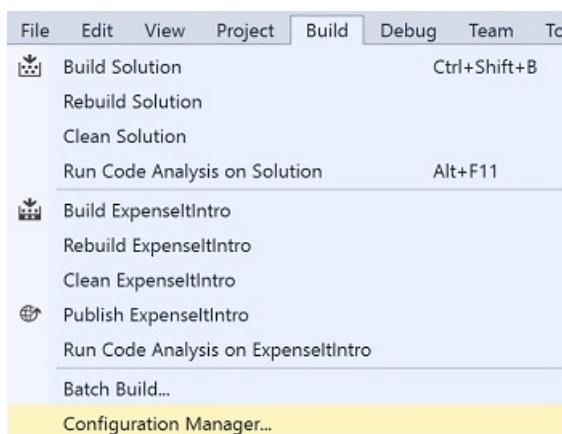
Create a custom build configuration

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](#), [Understand build platforms](#), and [How to: Set debug and release configurations](#).

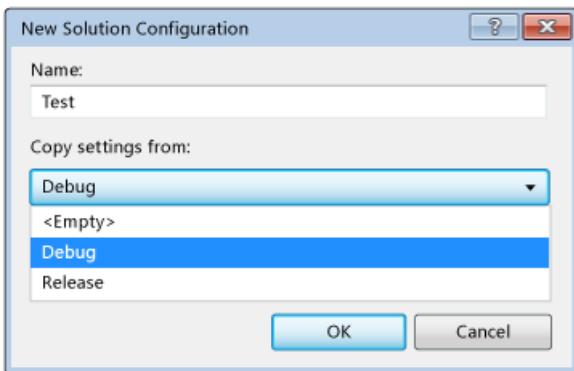
You can change or create configurations and platform settings by using the **Configuration Manager** dialog box. In this procedure, you'll create a build configuration for testing.

Create a build configuration

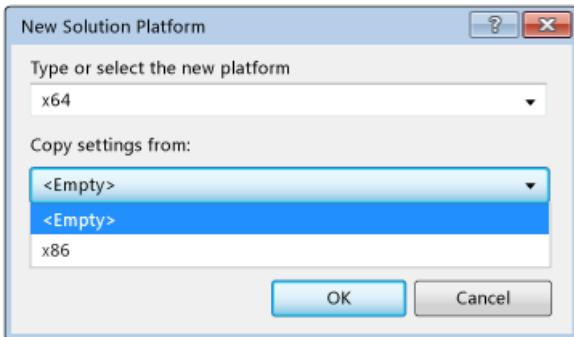
1. Open the **Configuration Manager** dialog box.



2. In the **Active solution configuration** list, choose <New...>.
3. 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.

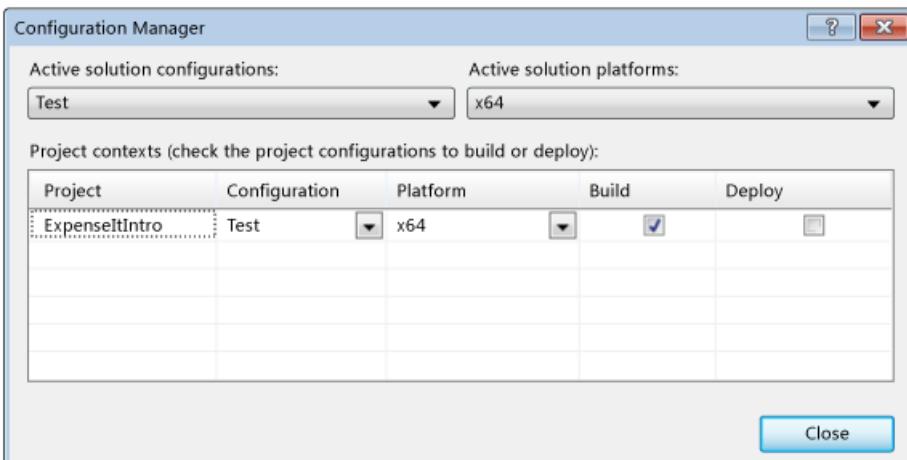


4. In the **Active solution platform** list, choose **<New...>**.
5. In the **New Solution Platform** dialog box, choose **x64**, and don't copy settings from the x86 platform.



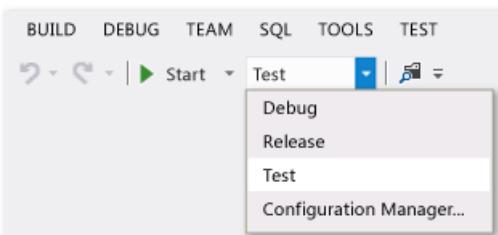
6. Choose the **OK** button.

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



7. 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**, or press **Ctrl+Shift+B**.

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.

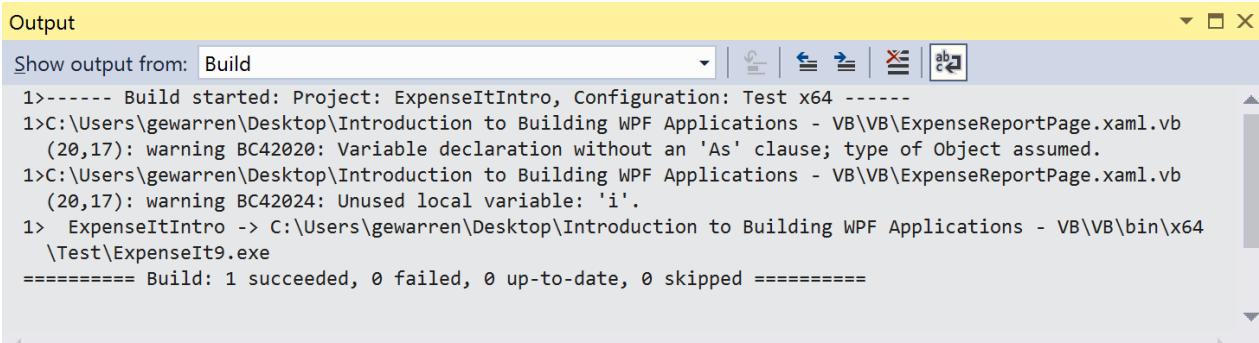
- In the C# project, open the *ExpenseReportPage.xaml.cs* file. In the **ExpenseReportPage** method, add the following code: `int i;`

OR

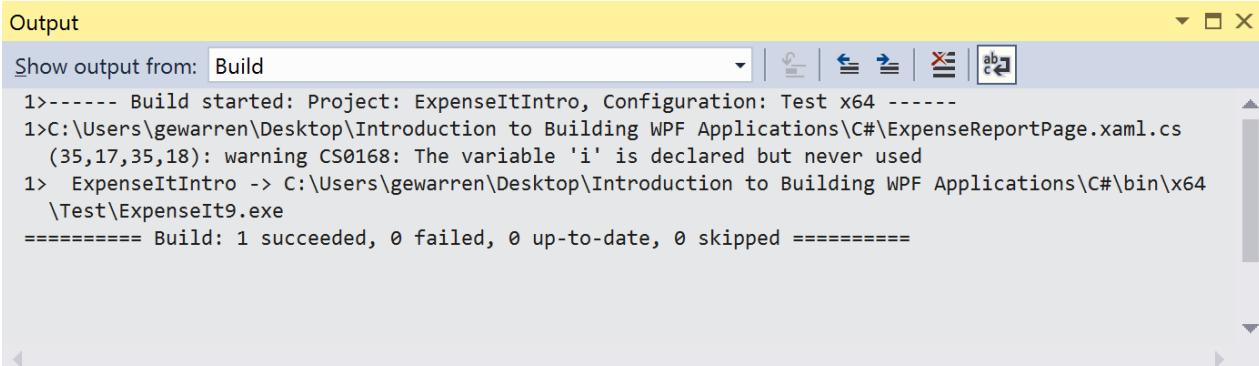
In the Visual Basic project, open the *ExpenseReportPage.xaml.vb* file. In the custom constructor **Public Sub New...**, add the following code: `Dim i`.

- Build the solution.

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



```
Output
Show output from: Build
1>----- Build started: Project: ExpenseItIntro, Configuration: Test x64 -----
1>C:\Users\gewarren\Desktop\Introduction to Building WPF Applications - VB\VB\ExpenseReportPage.xaml.vb
(20,17): warning BC42020: Variable declaration without an 'As' clause; type of Object assumed.
1>C:\Users\gewarren\Desktop\Introduction to Building WPF Applications - VB\VB\ExpenseReportPage.xaml.vb
(20,17): warning BC42024: Unused local variable: 'i'.
1> ExpenseItIntro -> C:\Users\gewarren\Desktop\Introduction to Building WPF Applications - VB\VB\bin\x64
\Test\ExpenseIt9.exe
===== Build: 1 succeeded, 0 failed, 0 up-to-date, 0 skipped ======
```



```
Output
Show output from: Build
1>----- Build started: Project: ExpenseItIntro, Configuration: Test x64 -----
1>C:\Users\gewarren\Desktop\Introduction to Building WPF Applications\C#\ExpenseReportPage.xaml.cs
(35,17,35,18): warning CS0168: The variable 'i' is declared but never used
1> ExpenseItIntro -> C:\Users\gewarren\Desktop\Introduction to Building WPF Applications\C#\bin\x64
\Test\ExpenseIt9.exe
===== Build: 1 succeeded, 0 failed, 0 up-to-date, 0 skipped ======
```

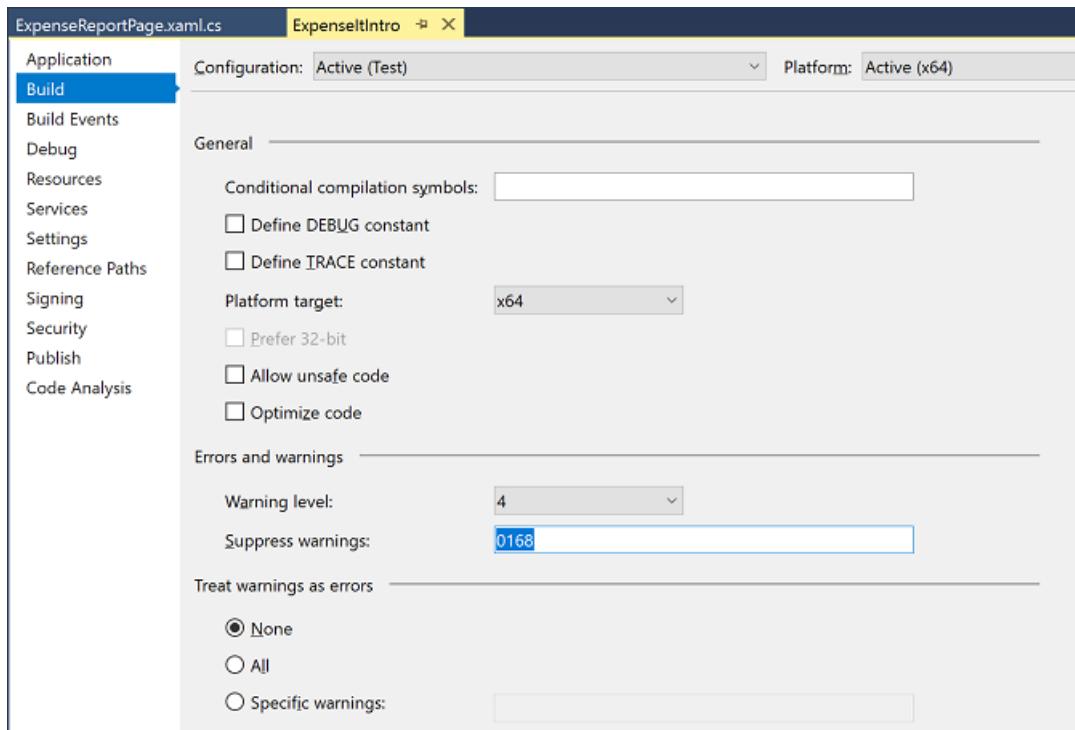
You can temporarily hide certain warning messages during a build rather than have them clutter up the build output.

Hide a specific C# warning

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

The **Project Designer** opens.

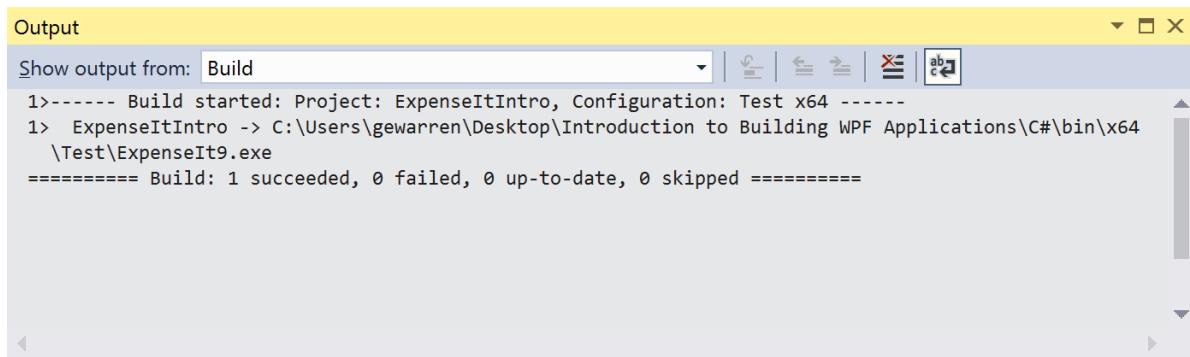
- Choose the **Build** page and then, in the **Suppress warnings** box, specify the warning number **0168**.



For more information, see [Build Page, Project Designer \(C#\)](#).

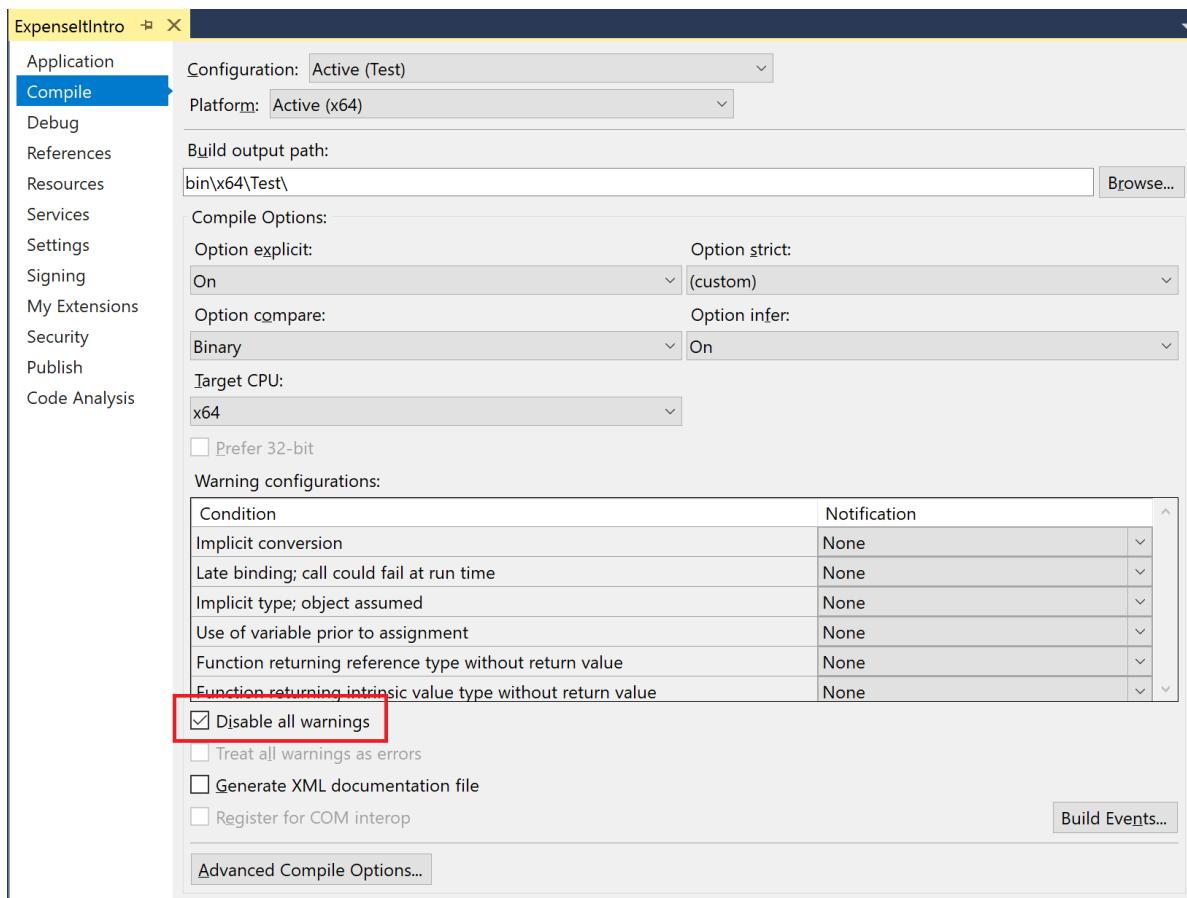
4. Build the solution.

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



Suppress all Visual Basic build warnings

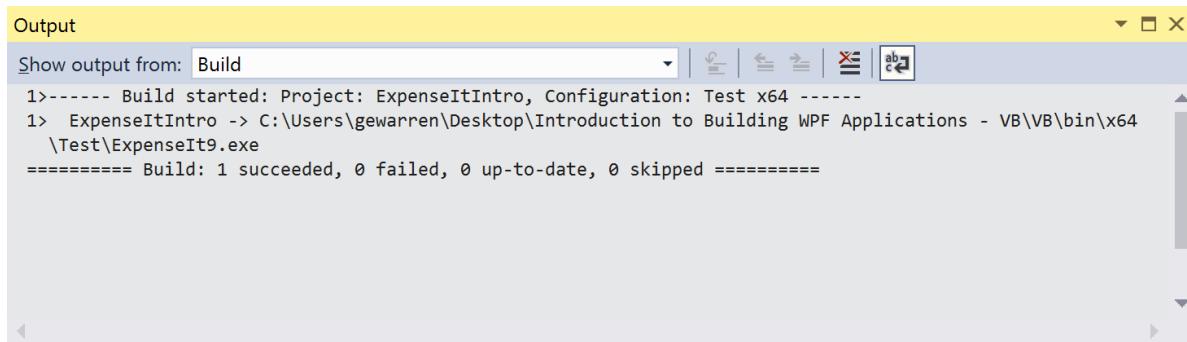
1. In **Solution Explorer**, choose the top-level project node.
2. On the menu bar, choose **View > Property Pages**.
The **Project Designer** opens.
3. On the **Compile** page, select the **Disable all warnings** check box.



For more information, see [Configure warnings in Visual Basic](#).

4. Build the solution.

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



For more information, see [How to: Suppress compiler warnings](#).

Display additional build details in the Output window

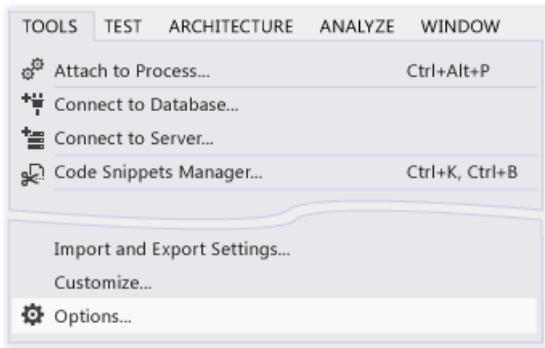
You can change how much information about the build process appears in the **Output** window. Build verbosity is usually set to **Minimal**, which means that the **Output** window displays only a summary of the build process along with any high priority warnings or errors. You can display more information about the build by using the [Options dialog box](#), [Projects and Solutions](#), [Build and Run](#).

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.



2. Choose the **Projects and Solutions** category, and then choose the **Build and Run** page.
3. In the **MSBuild project build output verbosity** list, choose **Normal**, and then choose the **OK** button.
4. On the menu bar, choose **Build > Clean Solution**.
5. 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` option lists the warning code, **1762**, that you specified earlier in this topic, along with three other warnings.

In the Visual Basic build, `/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](#).

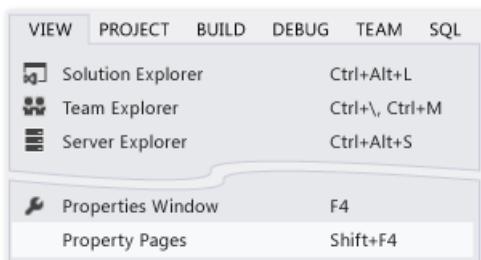
Create a Release Build

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](#) and [Build and clean projects and solutions in Visual Studio](#).

Specify a release build for Visual Basic

1. Open the **Project Designer**.



2. Choose the **Compile** page.
3. In the **Configuration** list, choose **Release**.
4. In the **Platform** list, choose **x86**.

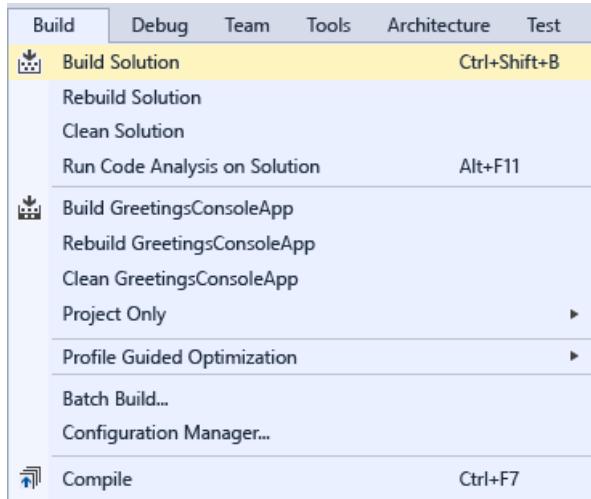
5. In the **Build output path** box, specify a network path.

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

IMPORTANT

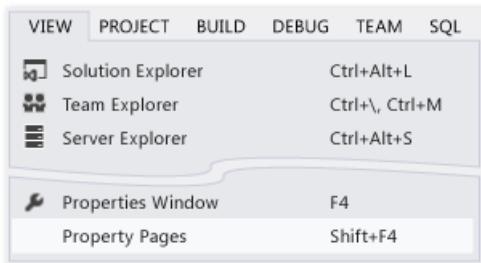
A message box might appear, warning you that the network share that you've specified might not be a trusted location. If you trust the location that you've specified, choose the **OK** button in the message box.

6. Build the application.



Specify a release build for C#

1. Open the **Project Designer**.



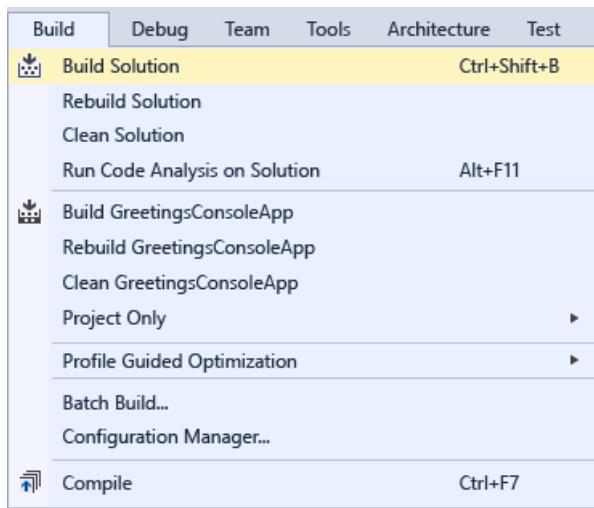
2. Choose the **Build** page.
3. In the **Configuration** list, choose **Release**.
4. In the **Platform** list, choose **x86**.
5. In the **Output path** box, specify a network path.

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.

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



The executable file is copied to the network path that you specified. Its path would be

`\myserver\builds\FileName.exe`.

Congratulations! You've successfully completed this walkthrough.

See also

- [Walkthrough: Build a project \(C++\)](#)
- [ASP.NET web application project precompilation overview](#)
- [Walkthrough: Use MSBuild](#)

Build and clean projects and solutions in Visual Studio

7/23/2019 • 3 minutes to read • [Edit Online](#)

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](#).

NOTE

This topic applies to Visual Studio on Windows. For Visual Studio for Mac, see [Build and clean projects and solutions in Visual Studio for Mac](#).

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.
3. 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**.

See also

- [How to: View, save, and configure build log files](#)
- [Obtaining build logs](#)
- [Compiling and building](#)
- [Understanding build configurations](#)

- [How to: Set debug and release configurations](#)
- [C/C++ building reference](#)
- [Devenv command line switches](#)
- [Solutions and projects](#)
- [Build and clean projects and solutions \(Visual Studio for Mac\)](#)

How to: Change the build output directory

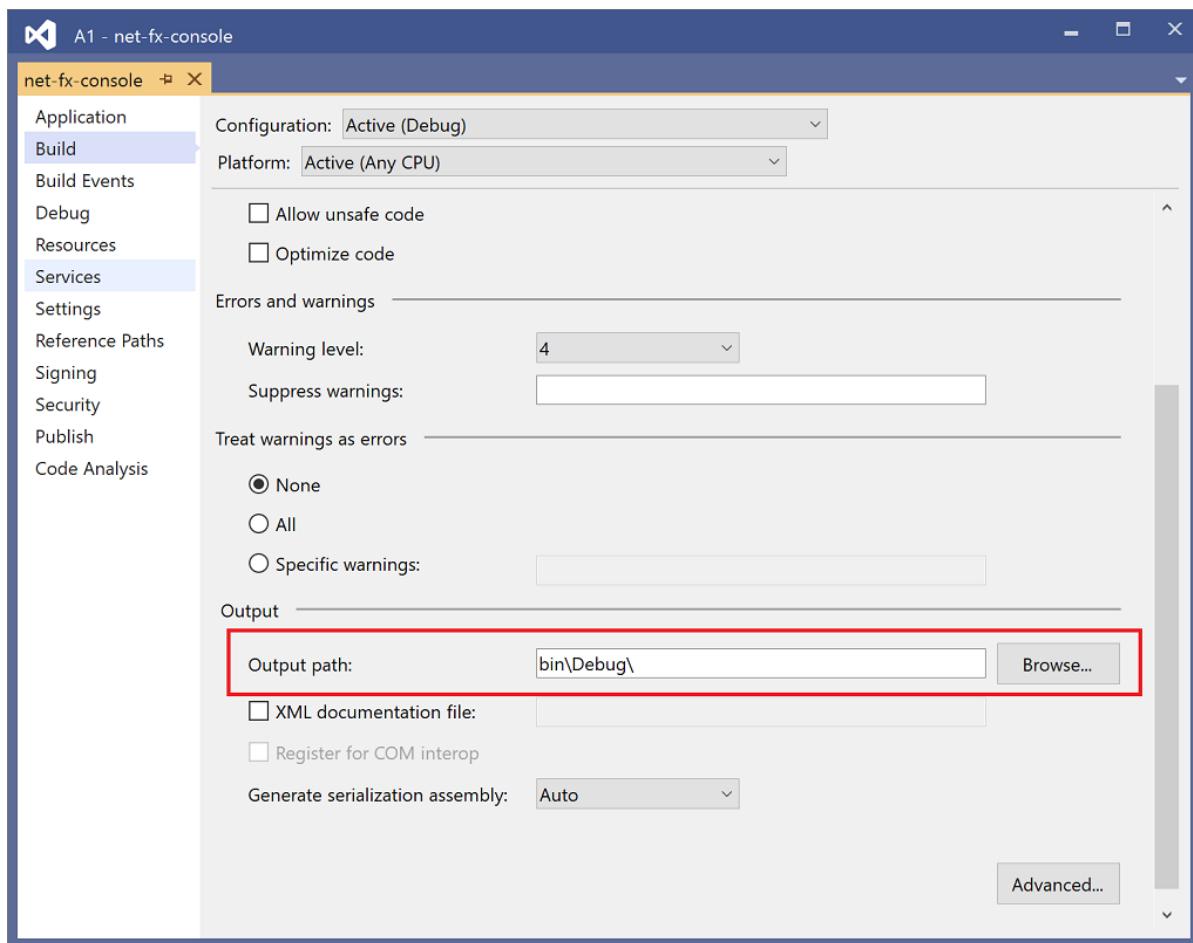
9/24/2019 • 2 minutes to read • [Edit Online](#)

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

Change the build output directory

1. To open the project's property pages, right-click on the project node in **Solution Explorer** and select **Properties**.
2. Select the appropriate tab based on your project type:
 - For C#, select the **Build** tab.
 - For Visual Basic, select the **Compile** tab.
 - For C++ or JavaScript, 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 Configurations**).
4. Find the output path entry on the page—it differs depending on your project type:
 - **Output path** for C# and JavaScript projects
 - **Build output path** for Visual Basic projects
 - **Output directory** for Visual C++ projects

Type in the path to generate output to (absolute or relative to the root project directory), or choose **Browse** to browse to that folder instead.



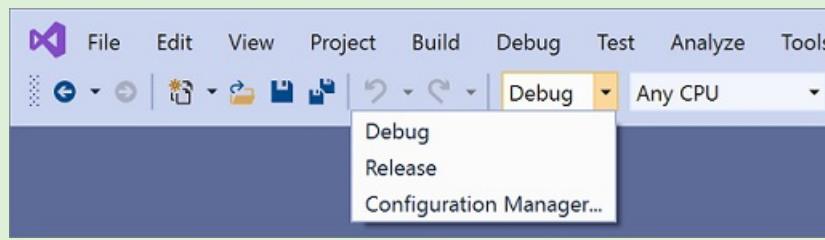
NOTE

Some projects will by default include framework and runtime in the build path. To change this, right-click the project node in **Solution Explorer**, select **Edit Project File**, and add the following:

```
<PropertyGroup>
  <AppendTargetFrameworkToOutputPath>false</AppendTargetFrameworkToOutputPath>
  <AppendRuntimeIdentifierToOutputPath>false</AppendRuntimeIdentifierToOutputPath>
</PropertyGroup>
```

TIP

If the output is not being generated to the location that you specified, make sure you're building the corresponding configuration (for example, **Debug** or **Release**) by selecting it on the menu bar of Visual Studio.



See also

- [Build page, Project Designer \(C#\)](#)
- [General Property page \(project\)](#)
- [Compile and build](#)

How to: Build to a common output directory

7/23/2019 • 2 minutes to read • [Edit Online](#)

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.

See also

- [Compile and build](#)
- [How to: Change the build output directory](#)

Specify custom build events in Visual Studio

8/9/2019 • 2 minutes to read • [Edit Online](#)

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 \(Visual Basic\)](#).
- C# and F#--[How to: Specify build events \(C#\)](#).
- Visual C++--[Specify 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](#).

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.

See also

- [Compile and build](#)
- [Pre-build Event/Post-build Event command line dialog box](#)
- [MSBuild special characters](#)
- [Walkthrough: Build an application](#)

How to: Set multiple startup projects

7/23/2019 • 2 minutes to read • [Edit Online](#)

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.

See also

- [Compile and build](#)
- [Create solutions and projects](#)
- [Manage project and solution properties](#)

How to: Create and remove project dependencies

7/23/2019 • 2 minutes to read • [Edit Online](#)

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.

3. On the **Dependencies** tab, select a project from the **Project** drop-down menu.
4. 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.

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

See also

- [Build and clean projects and solutions in Visual Studio](#)
- [Compile and build](#)
- [Understand build configurations](#)
- [Manage project and solution properties](#)

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

8/29/2019 • 2 minutes to read • [Edit Online](#)

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 click in the build output window and press **Ctrl+S**. Visual Studio prompts you for a location to save the information from the **Output** window into a .txt file.

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](#).

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, click somewhere in the text.
3. Press **Ctrl+S**.

Visual Studio prompts you for a location to save the build output.

You can also generate logs by running MSBuild directly from the command line, using the `-fileLogger` (`-fl`) command-line option. See [Obtain build logs with MSBuild](#).

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.

VERBOSITY LEVEL	DESCRIPTION
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.

For more information, see [Options dialog box, Projects and Solutions, Build and Run](#) and [LoggerVerbosity](#).

IMPORTANT

You must rebuild the project for your changes to take effect in the **Output** window (all projects) and the `<ProjectName>.txt` file (C++ projects only).

Use binary logs to make it easier to browse large log files

Binary logs are an optional feature for .NET projects that lets you have a richer log browsing experience that might make it easier to find information in large logs. To use binary logs, install the [Project System Tools](#). For more information, see <https://msbuildlog.com> and [Binary Log](#)

See also

- [Build and clean projects and solutions in Visual Studio](#)
- [Compile and build](#)

How to: Exclude projects from a build

11/7/2019 • 2 minutes to read • [Edit Online](#)

You can build a solution without building all projects that it contains. For example, you might exclude a project that breaks the build. You could then build the project after you investigate and address the issues.

You can exclude a project by taking the following approaches:

- Removing it temporarily from the active solution configuration.
- Creating a solution configuration that doesn't include the project.

For more information, see [Understand build configurations](#).

To temporarily remove a project from the active solution configuration

1. On the menu bar, choose **Build > Configuration Manager**.
2. In the **Project contexts** table, locate the project you want to exclude from the build.
3. In the **Build** column for the project, clear the check box.
4. Choose the **Close** button, and then rebuild the solution.

To create a solution configuration that excludes a project

1. On the menu bar, choose **Build > Configuration Manager**.
2. In the **Active solution configuration** list, choose **<New>**.
3. In the **Name** box, enter a name for the solution configuration.
4. In the **Copy settings from** list, choose the solution configuration on which you want to base the new configuration (for example, **Debug**), and then choose the **OK** button.
5. In the **Configuration Manager** dialog box, clear the check box in the **Build** column for the project that you want to exclude, and then choose the **Close** button.
6. On the **Standard** toolbar, verify that the new solution configuration is the active configuration in the **Solution Configurations** box.
7. On the menu bar, choose **Build > Rebuild Solution**.

Skipped projects

Projects can be skipped during the build because they are not up-to-date or because they are excluded from the configuration. Visual Studio uses MSBuild to build your projects. MSBuild only builds a target if the output is older than the input, as determined by the file timestamps. To force a rebuild, use the command **Build > Rebuild Solution**.

In the **Build** pane of the **Output** window, Visual Studio reports the number of projects that were up to date, the number that built successfully, the number that failed, and the number that were skipped. The skipped count does not include projects that were not built because they were up-to-date. When projects are excluded from the active configuration, they are skipped during the build. In the build output, you see a message indicating that the project is skipped:

```
2>----- Skipped Build: Project: ConsoleApp2, Configuration: Debug x86 -----  
2>Project not selected to build for this solution configuration
```

To find out why a project was skipped, note the active configuration (`Debug x86` in the previous example), and choose **Build > Configuration Manager**. You can view or change which projects are skipped for each configuration, as discussed in this article.

See also

- [Understand build configurations](#)
- [How to: Create and edit configurations](#)
- [How to: Build multiple configurations simultaneously](#)

How to: Suppress compiler warnings

10/21/2019 • 3 minutes to read • [Edit Online](#)

You can declutter a build log by filtering out one or more kinds of compiler warnings. For example, you might want to review only some of the output that's generated when you set the build log verbosity to **Normal**, **Detailed**, or **Diagnostic**. For more information about verbosity, see [How to: View, save, and configure build log files](#).

Suppress specific warnings for Visual C# or F#

Use the **Build** property page to suppress specific warnings for C# and F# projects.

1. In **Solution Explorer**, choose the project in which you want to suppress warnings.
2. On the menu bar, choose **View > Property Pages**.
3. Choose the **Build** page.
4. In the **Suppress warnings** box, specify the error codes of the warnings that you want to suppress, separated by semicolons.
5. Rebuild the solution.

Suppress specific warnings for C++

Use the **Configuration Properties** property page to suppress specific warnings for C++ projects.

1. In **Solution Explorer**, choose the project or source file in which you want to suppress warnings.
2. On the menu bar, choose **View > Property Pages**.
3. Choose the **Configuration Properties** category, choose the **C/C++** category, and then choose the **Advanced** page.
4. Perform one of the following steps:
 - In the **Disable Specific Warnings** box, specify the error codes of the warnings that you want to suppress, separated by a semicolon.
 - In the **Disable Specific Warnings** box, choose **Edit** to display more options.
5. Choose the **OK** button, and then rebuild the solution.

Suppress warnings for Visual Basic

You can hide specific compiler warnings for Visual Basic by editing the `.vbproj` file for the project. To suppress warnings by *category*, you can use the [Compile property page](#). For more information, see [Configure warnings in Visual Basic](#).

To suppress specific warnings for Visual Basic

This example shows you how to edit the `.vbproj` file to suppress specific compiler warnings.

1. In **Solution Explorer**, choose the project in which you want to suppress warnings.
2. On the menu bar, choose **Project > Unload Project**.
3. In **Solution Explorer**, open the right-click or shortcut menu for the project, and then choose **Edit**

<ProjectName>.vbproj.

The XML project file opens in the code editor.

- Locate the `<NoWarn>` element for the build configuration you're building with, and add one or more warning numbers as the value of the `<NoWarn>` element. If you specify multiple warning numbers, separate them with a comma.

The following example shows the `<NoWarn>` element for the *Debug* build configuration on an x86 platform, with two compiler warnings suppressed:

```
<PropertyGroup Condition=" '$(Configuration)|$(Platform)' == 'Debug|x86' ">
  <PlatformTarget>x86</PlatformTarget>
  <DebugSymbols>true</DebugSymbols>
  <DebugType>full</DebugType>
  <Optimize>false</Optimize>
  <OutputPath>bin\Debug\</OutputPath>
  <DefineDebug>true</DefineDebug>
  <DefineTrace>true</DefineTrace>
  <ErrorReport>prompt</ErrorReport>
  <NoWarn>40059,42024</NoWarn>
  <WarningLevel>1</WarningLevel>
</PropertyGroup>
```

NOTE

.NET Core projects do not contain build configuration property groups by default. To suppress warnings in a .NET Core project, add the build configuration section to the file manually. For example:

```
<Project Sdk="Microsoft.NET.Sdk">
  <PropertyGroup>
    <OutputType>Exe</OutputType>
    <TargetFramework>netcoreapp2.0</TargetFramework>
    <RootNamespace>VBDotNetCore_1</RootNamespace>
  </PropertyGroup>
  <PropertyGroup Condition=" '$(Configuration)|$(Platform)' == 'Debug|AnyCPU' ">
    <NoWarn>42016,41999,42017</NoWarn>
  </PropertyGroup>
</Project>
```

- Save the changes to the `.vbproj` file.
- On the menu bar, choose **Project** > **Reload Project**.
- On the menu bar, choose **Build** > **Rebuild Solution**.

The **Output** window no longer shows the warnings that you specified.

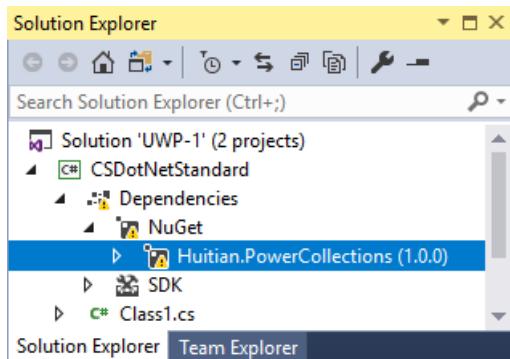
For more information, see the [/nowarn compiler option](#) for the Visual Basic command-line compiler.

Suppress warnings for NuGet packages

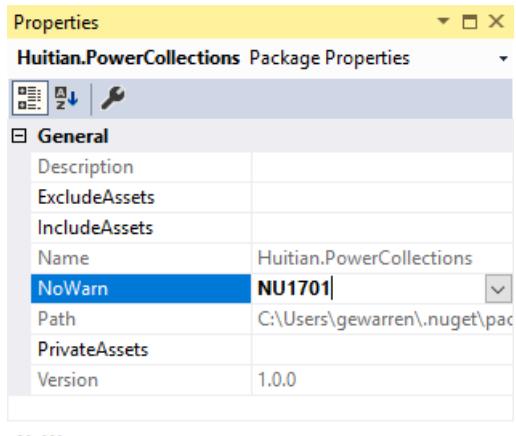
In some cases, you may want to suppress NuGet compiler warnings for a single NuGet package, instead of for an entire project. The warning serves a purpose, so you don't want to suppress it at the project level. For example, one of the NuGet warnings tells you that the package may not be fully compatible with your project. If you suppress it at the project level and later add an additional NuGet package, you would never know if it was producing the compatibility warning.

To suppress a specific warning for a single NuGet package

1. In **Solution Explorer**, select the NuGet package you want to suppress compiler warnings for.



2. From the right-click or context menu, select **Properties**.
3. In the **NoWarn** box of the package's properties, enter the warning number you want to suppress for this package. If you want to suppress more than one warning, use a comma to separate the warning numbers.



The warning disappears from **Solution Explorer** and the **Error List**.

See also

- [Walkthrough: Build an application](#)
- [How to: View, save, and configure build log files](#)
- [Compile and build](#)

Build actions

9/24/2019 • 2 minutes to read • [Edit Online](#)

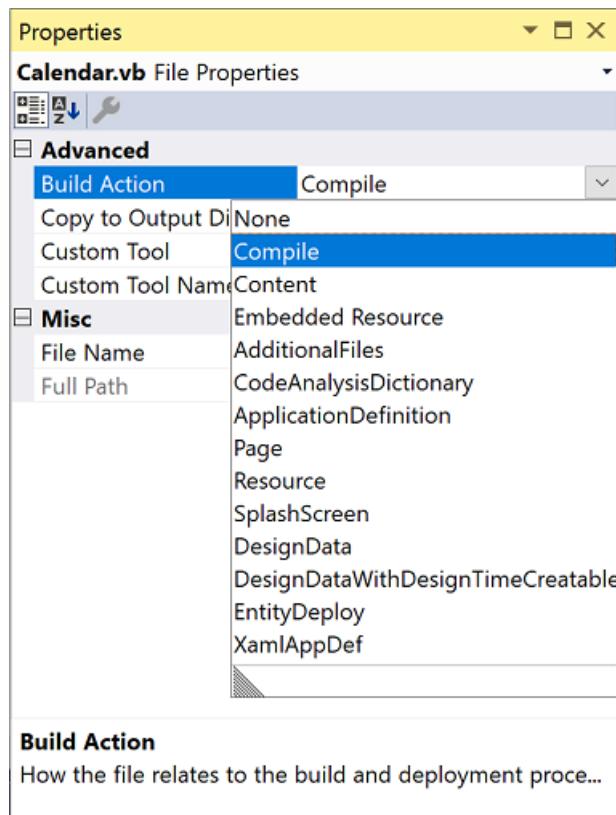
All files in a Visual Studio project have a build action. The build action controls what happens to the file when the project is compiled.

NOTE

This topic applies to Visual Studio on Windows. For Visual Studio for Mac, see [Build actions in Visual Studio for Mac](#).

Set a build action

To set the build action for a file, open the file's properties in the **Properties** window by selecting the file in **Solution Explorer** and pressing **Alt+Enter**. Or, right-click on the file in **Solution Explorer** and choose **Properties**. In the **Properties** window, under the **Advanced** section, use the drop-down list next to **Build Action** to set a build action for the file.



Build action values

Some of the more common build actions for C# and Visual Basic project files are:

BUILD ACTION	PROJECT TYPES	DESCRIPTION
--------------	---------------	-------------

BUILD ACTION	PROJECT TYPES	DESCRIPTION
AdditionalFiles	C#, Visual Basic	A non-source text file that's passed to the C# or Visual Basic compiler as input. This build action is mainly used to provide inputs to analyzers that are referenced by a project to verify code quality. For more information, see Use additional files .
ApplicationDefinition	WPF	The file that defines your application. When you first create a project, this is <i>App.xaml</i> .
CodeAnalysisDictionary	.NET	A custom word dictionary, used by Code Analysis for spell checking. See How to: Customize the Code Analysis Dictionary
Compile	any	The file is passed to the compiler as a source file.
Content	.NET	A file marked as Content can be retrieved as a stream by calling Application.GetContentStream . For ASP.NET projects, these files are included as part of the site when it's deployed.
DesignData	WPF	Used for XAML ViewModel files, to enable user controls to be viewed at design time, with dummy types and sample data.
DesignDataWithDesignTimeCreateable	WPF	Like DesignData , but with actual types.
Embedded Resource	.NET	The file is passed to the compiler as a resource to be embedded in the assembly. You can call System.Reflection.Assembly.GetManifestResourceStream to read the file from the assembly.
EntityDeploy	.NET	For Entity Framework (EF) .edmx files that specify deployment of EF artifacts.
Fakes	.NET	Used for the Microsoft Fakes testing framework. See Isolate code under test using Microsoft Fakes
None	any	The file isn't part of the build in any way. This value can be used for documentation files such as "ReadMe" files, for example.
Page	WPF	Compile a XAML file to a binary .bam! file for faster loading at run time.

BUILD ACTION	PROJECT TYPES	DESCRIPTION
Resource	WPF	Specifies to embed the file in an assembly manifest resource file with the extension <code>.g.resources</code> .
Shadow	.NET	Used for an <code>.accessor</code> file that contains a list of built assembly filenames, one per line. For each assembly on the list, generate public classes with the names <code>ClassName_Accessor</code> that are just like the originals, but with public methods instead of private methods. Used for unit testing.
Splash Screen	WPF	Specifies an image file to be displayed at run time when the app is starting up.
XamlAppDef	Windows Workflow Foundation	Instructs the build to build a workflow XAML file into an assembly with an embedded workflow.

NOTE

Additional build actions may be defined by for specific project types, so the list of build actions depends on the project type and values might appear that are not in this list.

See also

- [C# compiler options](#)
- [Visual Basic compiler options](#)
- [Build actions \(Visual Studio for Mac\)](#)

Understand build configurations

8/9/2019 • 4 minutes to read • [Edit Online](#)

You can store different configurations of solution and project properties to use in different kinds of builds. To create, select, modify, or delete a configuration, you can use the **Configuration Manager**. To open it, on the menu bar, choose **Build > Configuration Manager**, or just type **Configuration** in the search box. You can also use the **Solution Configurations** list on the **Standard** toolbar to select a configuration or open the **Configuration Manager**.

NOTE

This topic applies to Visual Studio on Windows. For Visual Studio for Mac, see [Build configurations in Visual Studio for Mac](#).

NOTE

If you can't find solution configuration settings on the toolbar and can't access the **Configuration Manager**, Visual Basic development settings may be applied. For more information, see [How to: Manage configurations with Visual Basic developer settings applied](#).

By default, Debug and Release configurations are included in projects that are created by using Visual Studio templates. A Debug configuration supports the debugging of an app, and a Release configuration builds a version of the app that can be deployed. For more information, see [How to: Set debug and release configurations](#). You can also create custom solution configurations and project configurations. For more information, see [How to: Create and edit configurations](#).

Solution configurations

A solution configuration specifies how projects in the solution are to be built and deployed. To modify a solution configuration or define a new one, in the **Configuration Manager**, under **Active solution configuration**, choose **Edit** or **New**.

Each entry in the **Project contexts** box in a solution configuration represents a project in the solution. For every combination of **Active solution configuration** and **Active solution platform**, you can set how each project is used. (For more information about solution platforms, see [Understand build platforms](#).)

NOTE

When you define a new solution configuration and select the **Create new project configurations** check box, Visual Studio automatically assigns the new configuration to all of the projects. Likewise, when you define a new solution platform and select the **Create new project platforms** check box, Visual Studio automatically assigns the new platform to all of the projects. Also, if you add a project that targets a new platform, Visual Studio adds that platform to the list of solution platforms and assigns it to all of the projects.

You can still modify the settings for each project.

The active solution configuration also provides context to the IDE. For example, if you're working on a project and the configuration specifies that it will be built for a mobile device, the **Toolbox** displays only items that can be used in a mobile device project.

Project configurations

The configuration and platform that a project targets are used together to specify the properties to use when it's built. A project can have a different set of property definitions for each combination of configuration and platform. To modify the properties of a project, you can use its Property Pages. (In **Solution Explorer**, open the shortcut menu for the project and then choose **Properties**.)

For each project configuration, you can define configuration-dependent properties as needed. For example, for a particular build, you can set which project items will be included, and what output files will be created, where they will be put, and how they will be optimized.

Project configurations can differ considerably. For example, the properties of one configuration might specify that its output file be optimized to occupy the minimum space, while another configuration might specify that its executable runs at the maximum speed.

Project configurations are stored by solution—not by user—so that they can be shared by a team.

Although project dependencies are configuration-independent, only the projects that are specified in the active solution configuration will be built.

How Visual Studio assigns project configurations

When you define a new solution configuration and don't copy settings from an existing one, Visual Studio uses the following criteria to assign default project configurations. The criteria are evaluated in the order shown.

1. If a project has a configuration name (*<configuration name> <platform name>*) that exactly matches the name of the new solution configuration, that configuration is assigned. Configuration names are not case-sensitive.
2. If the project has a configuration name in which the configuration-name part matches the new solution configuration, that configuration is assigned, whether the platform portion matches or not.
3. If there is still no match, the first configuration that's listed in the project is assigned.

How Visual Studio assigns solution configurations

When you create a project configuration (in the **Configuration Manager**, by choosing **New** on the drop-down menu in the **Configuration** column for that project) and select the **Create new solution configurations** check box, Visual Studio looks for a like-named solution configuration to build the project on each platform it supports. In some cases, Visual Studio renames existing solution configurations or defines new ones.

Visual Studio uses the following criteria to assign solution configurations.

- If a project configuration doesn't specify a platform or specifies just one platform, then a solution configuration whose name matches that of the new project configuration is either found or added. The default name of this solution configuration does not include a platform name; it takes the form *<project configuration name>*.
- If a project supports multiple platforms, a solution configuration is either found or added for each supported platform. The name of each solution configuration includes both the project configuration name and the platform name, and has the form *<project configuration name> <platform name>*.

See also

- [Walkthrough: Build an application](#)
- [Compile and build](#)
- [Solutions and projects](#)

- [C/C++ build reference](#)
- [Devenv command line switches](#)
- [Build configurations \(Visual Studio for Mac\)](#)

How to: Create and edit configurations

8/9/2019 • 3 minutes to read • [Edit Online](#)

You can create several build configurations for a solution. For example, you can configure a debug build that your testers can use to find and fix problems, and you can configure different kinds of builds that you can distribute to different customers.

NOTE

This topic applies to Visual Studio on Windows. For Visual Studio for Mac, see [Create and edit configurations in Visual Studio for Mac](#).

Create build configurations

You can use the **Configuration Manager** dialog box to select or modify existing build configurations, or to create new ones.

To open the **Configuration Manager** dialog box, in **Solution Explorer**, open the shortcut menu for the solution and then choose **Configuration Manager**.

NOTE

If the **Configuration Manager** command doesn't appear on the shortcut menu, look under the **Build** menu on the menu bar. If it doesn't appear there either, on the menu bar, choose **Tools > Options**, and then in the left pane of the **Options** dialog box, expand **Projects and Solutions > General**, and in the right pane, select the **Show advanced build configurations** check box.

In the **Configuration Manager** dialog box, you can use the **Active solution configuration** drop-down list to select a solution-wide build configuration, modify an existing one, or create a new configuration. You can use the **Active solution platform** drop-down list to select the platform that the configuration targets, modify an existing one, or add a new platform. The **Project contexts** pane lists the projects in the solution. For each project, you can select a project-specific configuration and platform, modify existing ones, or create a new configuration or add a new platform. You can also select check boxes that indicate whether each project is included when you use the solution-wide configuration to build or deploy the solution.

After you set up the configurations you want, you can set project properties that are appropriate for those configurations.

Set properties based on configurations

To set properties based on configurations, in **Solution Explorer**, open the shortcut menu for a project and then choose **Properties**. You can set properties for your configurations. For example, for a release configuration, you can specify that code is optimized when the solution is built, and for a debug configuration, you can specify that the `DEBUG` conditional compilation symbol is included.

For more information about property page settings, see [Manage project and solution properties](#).

Create a project configuration

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

2. Select a project in the **Project** column.
3. In the **Configuration** drop-down list for that project, choose **New**.
The **New Project Configuration** dialog box opens.
4. In the **Name** box, enter a name for the new configuration.
5. To use the property settings from an existing project configuration, in the **Copy settings from** drop-down list, choose a configuration.
6. To create a solution-wide configuration at the same time, select the **Create new solution configuration** check box.

Rename a project configuration

1. Open the **Configuration Manager** dialog box.
2. In the **Project** column, select the project that has the project configuration you want to rename.
3. In the **Configuration** drop-down list for that project, choose **Edit**.
The **Edit Project Configurations** dialog box opens.
4. Select the project configuration name you want to change.
5. Select **Rename**, and then enter a new name.

Create and modify solution-wide build configurations

To create a solution-wide build configuration

1. Open the **Configuration Manager** dialog box.
2. In the **Active solution configuration** drop-down list, choose **New**.
The **New Solution Configuration** dialog box opens.
3. In the **Name** text box, enter a name for the new configuration.
4. To use the settings from an existing solution configuration, in the **Copy settings from** drop-down list, choose a configuration.
5. If you want to create project configurations at the same time, select the **Create new project configurations** check box.

To rename a solution-wide build configuration

1. Open the **Configuration Manager** dialog box.
2. In the **Active solution configuration** drop-down list, choose **Edit**.

The **Edit Solution Configurations** dialog box opens.

3. Select the solution configuration name you want to change.
4. Select **Rename**, and then enter a new name.

To modify a solution-wide build configuration

1. Open the **Configuration Manager** dialog box.
2. In the **Active solution configuration** drop-down list, select the configuration you want.
3. In the **Project contexts** pane, for every project, select the **Configuration** and **Platform** you want, and

select whether to **Build** it and whether to **Deploy** it.

See also

- [Understand build configurations](#)
- [Build and clean projects and solutions in Visual Studio](#)
- [Manage project and solution properties](#)
- [Create and edit configurations \(Visual Studio for Mac\)](#)

How to: Manage build configurations with Visual Basic developer settings applied

7/23/2019 • 2 minutes to read • [Edit Online](#)

By default, all advanced build configuration options are hidden when Visual Basic developer settings are applied. This article explains how to manually enable these build settings.

Enable advanced build configurations

By default, the Visual Basic developer settings hide the option to open the **Configuration Manager** dialog box and the **Configuration** and **Platform** lists in the **Project Designer**.

1. On the **Tools** menu, click **Options**.
2. Expand **Projects and Solutions**, and click **General**.

NOTE

The **General** node is visible even if the **Show all settings** option is unchecked. If you want to see every option available, click **Show all settings**.

3. Click **Show advanced build configurations**.
4. Click **OK**.

Configuration Manager is now available on the **Build** menu, and the **Configuration** and **Platform** lists are visible in the **Project Designer**.

See also

- [Understand build configurations](#)
- [Compile and build](#)
- [Environment settings](#)

How to: Build multiple configurations simultaneously

7/23/2019 • 2 minutes to read • [Edit Online](#)

You can build most types of projects with multiple, or even all, of their build configurations at the same time by using the **Batch Build** dialog box. However, you can't build the following types of projects in multiple build configurations at the same time:

1. Windows 8.x Store apps built for Windows using JavaScript.
2. All Visual Basic projects.

For more information about build configurations, see [Understand build configurations](#).

To build a project in multiple build configurations

1. On the menu bar, choose **Build** > **Batch Build**.
2. In the **Build** column, select the check boxes for the configurations in which you want to build a project.

TIP

To edit or create a build configuration for a solution, choose **Build** > **Configuration Manager** on the menu bar to open the **Configuration Manager** dialog box. After you have edited a build configuration for a solution, choose the **Rebuild** button in the **Batch Build** dialog box to update all build configurations for the projects in the solution.

3. Choose the **Build** or **Rebuild** buttons to build the project with the configurations that you specified.

See also

- [How to: Create and edit configurations](#)
- [Understand build configurations](#)
- [Build multiple projects in parallel](#)

Understand build platforms

8/9/2019 • 2 minutes to read • [Edit Online](#)

You can store different versions of solution and project properties that apply to different target platforms. For example, you can create a Debug configuration that targets an x86 platform and a Debug configuration that targets an x64 platform. You can quickly change the active platform so that you can easily build multiple configurations.

In this section

[How to: Configure projects to target platforms](#)

Explains how to configure a project to target a specific platform.

[How to: Configure projects to target multiple platforms](#)

Explains how to configure a project to target multiple platforms.

See also

- [Walkthrough: Build an application](#)
- [Build and clean projects and solutions in Visual Studio](#)
- [Compile and build](#)

How to: Configure projects to target platforms

10/31/2019 • 3 minutes to read • [Edit Online](#)

Visual Studio enables you to set up your applications to target different platforms, including 64-bit platforms. For more information on 64-bit platform support in Visual Studio, see [64-bit applications](#).

Target platforms with the Configuration Manager

The **Configuration Manager** provides a way for you to quickly add a new platform to target with your project. If you select one of the platforms included with Visual Studio, the properties for your project are modified to build your project for the selected platform.

To configure a project to target a 64-bit platform

1. On the menu bar, choose **Build > Configuration Manager**.
2. In the **Active solution platform** list, choose a 64-bit platform for the solution to target, and then choose the **Close** button.
 - a. If the platform that you want doesn't appear in the **Active solution platform** list, choose **New**.
The **New Solution Platform** dialog box appears.
 - b. In the **Type or select the new platform** list, choose **x64**.

NOTE

If you give your configuration a new name, you may have to modify the settings in the **Project Designer** to target the correct platform.

- c. If you want to copy the settings from a current platform configuration, choose it, and then choose the **OK** button.

The properties for all projects that target the 64-bit platform are updated, and the next build of the project will be optimized for 64-bit platforms.

Target platforms in the Project Designer

The **Project Designer** also provides a way to target different platforms with your project. If selecting one of the platforms included in the list in the **New Solution Platform** dialog box does not work for your solution, you can create a custom configuration name and modify the settings in the **Project Designer** to target the correct platform.

Performing this task varies based on the programming language you are using. See the following links for more information:

- For Visual Basic projects, see [/platform \(Visual Basic\)](#).
- For Visual C# projects, see [Build page, Project Designer \(C#\)](#).
- For Visual C++ projects, see [/clr \(Common Language Runtime compilation\)](#).

Manually editing the project file

Sometimes, you need to manually edit the project file for some custom configuration. An example is when you have conditions that can't be specified in the IDE, such as a reference that is different for two different platforms, as in the following example.

Example: Referencing x86 and x64 assemblies and DLLs

You might have a .NET assembly or DLL that has both x86 and x64 versions. To set up your project to use these references, first add the reference, and then open the project file and edit it to add an `ItemGroup` With a condition that references both the configuration, and the target platform. For example, suppose the binary you are referencing is ClassLibrary1 and there are different paths for Debug and Release configurations, as well as x86 and x64 versions. Then, use four `ItemGroup` elements with all combinations of settings, as follows:

```
<Project Sdk="Microsoft.NET.Sdk">

<PropertyGroup>
  <OutputType>Exe</OutputType>
  <TargetFramework>netcoreapp2.0</TargetFramework>
  <Platforms>AnyCPU;x64;x86</Platforms>
</PropertyGroup>

<ItemGroup Condition=" '$(Configuration)|$(Platform)' == 'Debug|x64' " >
  <Reference Include="ClassLibrary1" >
    <HintPath>..\\ClassLibrary1\\ClassLibrary1\\bin\\x64\\Debug\\netstandard2.0\\ClassLibrary1.dll</HintPath>
  </Reference>
</ItemGroup>

<ItemGroup Condition=" '$(Configuration)|$(Platform)' == 'Release|x64' " >
  <Reference Include="ClassLibrary1" >
    <HintPath>..\\ClassLibrary1\\ClassLibrary1\\bin\\x64\\Release\\netstandard2.0\\ClassLibrary1.dll</HintPath>
  </Reference>
</ItemGroup>

<ItemGroup Condition=" '$(Configuration)|$(Platform)' == 'Debug|x86' " >
  <Reference Include="ClassLibrary1" >
    <HintPath>..\\ClassLibrary1\\ClassLibrary1\\bin\\x86\\Debug\\netstandard2.0\\ClassLibrary1.dll</HintPath>
  </Reference>
</ItemGroup>

<ItemGroup Condition=" '$(Configuration)|$(Platform)' == 'Release|x86' " >
  <Reference Include="ClassLibrary1" >
    <HintPath>..\\ClassLibrary1\\ClassLibrary1\\bin\\x86\\Release\\netstandard2.0\\ClassLibrary1.dll</HintPath>
  </Reference>
</ItemGroup>
</Project>
```

NOTE

In Visual Studio 2017, you need to unload the project before you can edit the project file. To unload the project, right-click on the project node, and choose **Unload project**. When done editing, save your changes and reload the project by right-clicking the project node and choosing **Reload project**.

For more information about the project file, see [MSBuild project file schema reference](#).

See also

- [Understand build platforms](#)
- [/platform \(C# compiler options\)](#)
- [64-bit applications](#)
- [Visual Studio IDE 64-Bit support](#)

- Understanding the project file

How to: Configure projects to target multiple platforms

7/23/2019 • 2 minutes to read • [Edit Online](#)

Visual Studio provides a way for a solution to target several different CPU architectures, or platforms, at once. The properties to set these are accessed through the **Configuration Manager** dialog box.

Target a platform

The **Configuration Manager** dialog box allows you to create and set solution-level and project-level configurations and platforms. Each combination of solution-level configurations and targets can have a unique set of properties associated with it, allowing you to easily switch between, for example, a release configuration that targets an x64 platform, a release configuration that targets an x86 platform, and a debug configuration that targets an x86 platform.

1. On the **Build** menu, click **Configuration Manager**.
2. In the **Active solution platform box**, select the platform you want your solution to target, or select **<New>** to create a new platform. Visual Studio will compile your application to target the platform that is set as the active platform in the **Configuration Manager** dialog box.

Remove a platform

If you realize that you have no need for a platform, you can remove it using the **Configuration Manager** dialog box. This will remove all solution and project settings that you configured for that combination of configuration and target.

1. On the **Build** menu, click **Configuration Manager**.
2. In the **Active solution platform box**, select **<Edit>**. The **Edit Solution Platforms** dialog box opens.
3. Click the platform you want to remove, and click **Remove**.

Target multiple platforms with one solution

Because you can change the settings based on the combination of configuration and platform settings, you can set up a solution that can target more than one platform.

To target multiple platforms

1. Use the **Configuration Manager** to add at least two target platforms for the solution.
2. Select the platform you want to target from the **Active solution platform** list.
3. Build the solution.

To build multiple solution configurations at once

1. Use the **Configuration Manager** to add at least two target platforms for the solution.
2. Use the **Batch Build** window to build several solution configurations at once.

It is possible to have a solution-level platform set to, for example, x64, and have no projects within that solution targeting the same platform. It is also possible to have multiple projects in your solution, each targeting different platforms. It is recommended that if you have one of these situations, you create a new

configuration with a descriptive name to avoid confusion.

See also

- [How to: Create and edit configurations](#)
- [Understand build configurations](#)
- [Build and clean projects and solutions in Visual Studio](#)

MSBuild

10/21/2019 • 9 minutes to read • [Edit Online](#)

The Microsoft Build Engine is a platform for building applications. This engine, which is also known as MSBuild, provides an XML schema for a project file that controls how the build platform processes and builds software. Visual Studio uses MSBuild, but it doesn't depend on Visual Studio. By invoking `msbuild.exe` on your project or solution file, you can orchestrate and build products in environments where Visual Studio isn't installed.

Visual Studio uses MSBuild to load and build managed projects. The project files in Visual Studio (`.csproj`, `.vbproj`, `.vcxproj`, and others) contain MSBuild XML code that executes when you build a project by using the IDE. Visual Studio projects import all the necessary settings and build processes to do typical development work, but you can extend or modify them from within Visual Studio or by using an XML editor.

For information about MSBuild for C++, see [MSBuild \(C++\)](#).

The following examples illustrate when you might run builds by using an MSBuild command line instead of the Visual Studio IDE.

- Visual Studio isn't installed. ([download MSBuild without Visual Studio](#))
- You want to use the 64-bit version of MSBuild. This version of MSBuild is usually unnecessary, but it allows MSBuild to access more memory.
- You want to run a build in multiple processes. However, you can use the IDE to achieve the same result on projects in C++ and C#.
- You want to modify the build system. For example, you might want to enable the following actions:
 - Preprocess files before they reach the compiler.
 - Copy the build outputs to a different place.
 - Create compressed files from build outputs.
 - Do a post-processing step. For example, you might want to stamp an assembly with a different version.

You can write code in the Visual Studio IDE but run builds by using MSBuild. As another alternative, you can build code in the IDE on a development computer but use an MSBuild command line to build code that's integrated from multiple developers.

NOTE

You can use Team Foundation Build to automatically compile, test, and deploy your application. Your build system can automatically run builds when developers check in code (for example, as part of a Continuous Integration strategy) or according to a schedule (for example, a nightly Build Verification Test build). Team Foundation Build compiles your code by using MSBuild. For more information, see [Azure Pipelines](#).

This topic provides an overview of MSBuild. For an introductory tutorial, see [Walkthrough: Using MSBuild](#).

Use MSBuild at a command prompt

To run MSBuild at a command prompt, pass a project file to `MSBuild.exe`, together with the appropriate command-line options. Command-line options let you set properties, execute specific targets, and set other options that

control the build process. For example, you would use the following command-line syntax to build the file `MyProj.proj` with the `Configuration` property set to `Debug`.

```
MSBuild.exe MyProj.proj -property:Configuration=Debug
```

For more information about MSBuild command-line options, see [Command-line reference](#).

IMPORTANT

Before you download a project, determine the trustworthiness of the code.

Project file

MSBuild uses an XML-based project file format that's straightforward and extensible. The MSBuild project file format lets developers describe the items that are to be built, and also how they are to be built for different operating systems and configurations. In addition, the project file format lets developers author reusable build rules that can be factored into separate files so that builds can be performed consistently across different projects in the product.

The following sections describe some of the basic elements of the MSBuild project file format. For a tutorial about how to create a basic project file, see [Walkthrough: Creating an MSBuild project file from scratch](#).

Properties

Properties represent key/value pairs that can be used to configure builds. Properties are declared by creating an element that has the name of the property as a child of a `PropertyGroup` element. For example, the following code creates a property named `BuildDir` that has a value of `Build`.

```
<PropertyGroup>
  <BuildDir>Build</BuildDir>
</PropertyGroup>
```

You can define a property conditionally by placing a `Condition` attribute in the element. The contents of conditional elements are ignored unless the condition evaluates to `true`. In the following example, the `Configuration` element is defined if it hasn't yet been defined.

```
<Configuration Condition="$(Configuration) == ''>Debug</Configuration>
```

Properties can be referenced throughout the project file by using the syntax `$(<PropertyName>)`. For example, you can reference the properties in the previous examples by using `$(BuildDir)` and `$(Configuration)`.

For more information about properties, see [MSBuild properties](#).

Items

Items are inputs into the build system and typically represent files. Items are grouped into item types, based on user-defined item names. These item types can be used as parameters for tasks, which use the individual items to perform the steps of the build process.

Items are declared in the project file by creating an element that has the name of the item type as a child of an `ItemGroup` element. For example, the following code creates an item type named `Compile`, which includes two files.

```
<ItemGroup>
  <Compile Include = "file1.cs"/>
  <Compile Include = "file2.cs"/>
</ItemGroup>
```

Item types can be referenced throughout the project file by using the syntax @(<ItemType>). For example, the item type in the example would be referenced by using `@(Compile)`.

In MSBuild, element and attribute names are case-sensitive. However, property, item, and metadata names are not. The following example creates the item type `Compile`, `comPile`, or any other case variation, and gives the item type the value "one.cs;two.cs".

```
<ItemGroup>
  <Compile Include="one.cs" />
  <comPile Include="two.cs" />
</ItemGroup>
```

Items can be declared by using wildcard characters and may contain additional metadata for more advanced build scenarios. For more information about items, see [Items](#).

Tasks

Tasks are units of executable code that MSBuild projects use to perform build operations. For example, a task might compile input files or run an external tool. Tasks can be reused, and they can be shared by different developers in different projects.

The execution logic of a task is written in managed code and mapped to MSBuild by using the [UsingTask](#) element. You can write your own task by authoring a managed type that implements the [ITask](#) interface. For more information about how to write tasks, see [Task writing](#).

MSBuild includes common tasks that you can modify to suit your requirements. Examples are [Copy](#), which copies files, [MakeDir](#), which creates directories, and [Csc](#), which compiles Visual C# source code files. For a list of available tasks together with usage information, see [Task reference](#).

A task is executed in an MSBuild project file by creating an element that has the name of the task as a child of a [Target](#) element. Tasks typically accept parameters, which are passed as attributes of the element. Both MSBuild properties and items can be used as parameters. For example, the following code calls the [MakeDir](#) task and passes it the value of the `BuildDir` property that was declared in the earlier example.

```
<Target Name="MakeBuildDirectory">
  <MakeDir Directories="$(BuildDir)" />
</Target>
```

For more information about tasks, see [Tasks](#).

Targets

Targets group tasks together in a particular order and expose sections of the project file as entry points into the build process. Targets are often grouped into logical sections to increase readability and to allow for expansion. Breaking the build steps into targets lets you call one piece of the build process from other targets without copying that section of code into every target. For example, if several entry points into the build process require references to be built, you can create a target that builds references and then run that target from every entry point where it's required.

Targets are declared in the project file by using the [Target](#) element. For example, the following code creates a target named `Compile`, which then calls the [Csc](#) task that has the item list that was declared in the earlier example.

```
<Target Name="Compile">
  <Csc Sources="@((Compile))" />
</Target>
```

In more advanced scenarios, targets can be used to describe relationships among one another and perform dependency analysis so that whole sections of the build process can be skipped if that target is up-to-date. For more information about targets, see [Targets](#).

Build logs

You can log build errors, warnings, and messages to the console or another output device. For more information, see [Obtaining build logs](#) and [Logging in MSBuild](#).

Use MSBuild in Visual Studio

Visual Studio uses the MSBuild project file format to store build information about managed projects. Project settings that are added or changed by using the Visual Studio interface are reflected in the `.*proj` file that's generated for every project. Visual Studio uses a hosted instance of MSBuild to build managed projects. This means that a managed project can be built in Visual Studio or at a command prompt (even if Visual Studio isn't installed), and the results will be identical.

For a tutorial about how to use MSBuild in Visual Studio, see [Walkthrough: Using MSBuild](#).

Multitargeting

By using Visual Studio, you can compile an application to run on any one of several versions of the .NET Framework. For example, you can compile an application to run on the .NET Framework 2.0 on a 32-bit platform, and you can compile the same application to run on the .NET Framework 4.5 on a 64-bit platform. The ability to compile to more than one framework is named multitargeting.

These are some of the benefits of multitargeting:

- You can develop applications that target earlier versions of the .NET Framework, for example, versions 2.0, 3.0, and 3.5.
- You can target frameworks other than the .NET Framework, for example, Silverlight.
- You can target a *framework profile*, which is a predefined subset of a target framework.
- If a service pack for the current version of the .NET Framework is released, you could target it.
- Multitargeting guarantees that an application uses only the functionality that's available in the target framework and platform.

For more information, see [Multitargeting](#).

See also

TITLE	DESCRIPTION
Walkthrough: Creating an MSBuild project file from scratch	Shows how to create a basic project file incrementally, by using only a text editor.
Walkthrough: Using MSBuild	Introduces the building blocks of MSBuild and shows how to write, manipulate, and debug MSBuild projects without closing the Visual Studio IDE.

TITLE	DESCRIPTION
MSBuild concepts	Presents the four building blocks of MSBuild: properties, items, targets, and tasks.
Items	Describes the general concepts behind the MSBuild file format and how the pieces fit together.
MSBuild properties	Introduces properties and property collections. Properties are key/value pairs that can be used to configure builds.
Targets	Explains how to group tasks together in a particular order and enable sections of the build process to be called on the command line.
Tasks	Shows how to create a unit of executable code that can be used by MSBuild to perform atomic build operations.
Conditions	Discusses how to use the <code>Condition</code> attribute in an MSBuild element.
Advanced concepts	Presents batching, performing transforms, multitargeting, and other advanced techniques.
Logging in MSBuild	Describes how to log build events, messages, and errors.
Additional resources	Lists community and support resources for more information about MSBuild.

Reference

- [MSBuild reference](#) Links to topics that contain reference information.
- [Glossary](#) Defines common MSBuild terms.

How to: Specify build events (Visual Basic)

7/24/2019 • 4 minutes to read • [Edit Online](#)

Build events in Visual Basic can be used to run scripts, macros, or other actions as a part of the compilation process. Pre-build events occur before compilation; post-build events occur after compilation.

Build events are specified in the **Build Events** dialog box, available from the **Compile** page of the **Project Designer**.

NOTE

Visual Basic Express does not support entry of build events. This is supported only in the full Visual Studio product.

How to specify pre-build and post-build events

To specify a build event

1. With a project selected in **Solution Explorer**, on the **Project** menu, click **Properties**.
2. Click the **Compile** tab.
3. Click the **Build Events** button to open the **Build Events** dialog box.
4. Enter the command-line arguments for your pre-build or post-build action, and then click **OK**.

NOTE

Add a `call` statement before all post-build commands that run `.bat` files. For example, `call C:\MyFile.bat` or `call C:\MyFile.bat call C:\MyFile2.bat`.

NOTE

If your pre-build or post-build event does not complete successfully, you can terminate the build by having your event action exit with a code other than zero (0), which indicates a successful action.

Example: How to change manifest information using a post-build event

The following procedure shows how to set the minimum operating system version in the application manifest using an `.exe` command called from a post-build event (the `.exe.manifest` file in the project directory). The minimum operating system version is a four-part number such as 4.10.0.0. To do this, the command will change the `<dependentOS>` section of the manifest:

```
<dependentOS>
  <osVersionInfo>
    <os majorVersion="4" minorVersion="10" buildNumber="0" servicePackMajor="0" />
  </osVersionInfo>
</dependentOS>
```

To create an `.exe` command to change the application manifest

1. Create a console application for the command. From the **File** menu, click **New**, and then click **Project**.
2. In the **New Project** dialog box, in the **Visual Basic** node, select **Windows** and then the **Console Application** template. Name the project `ChangeOSVersionVB`.
3. In *Module1.vb*, add the following line to the other `Imports` statements at the top of the file:

```
Imports System.Xml
```

4. Add the following code in `Sub Main`:

```
Sub Main()
    Dim applicationManifestPath As String
    applicationManifestPath = My.Application.CommandLineArgs(0)
    Console.WriteLine("Application Manifest Path: " & applicationManifestPath.ToString)

    'Get version name
    Dim osVersion As Version
    If My.Application.CommandLineArgs.Count >= 2 Then
        osVersion = New Version(My.Application.CommandLineArgs(1).ToString)
    Else
        Throw New ArgumentException("OS Version not specified.")
    End If
    Console.WriteLine("Desired OS Version: " & osVersion.ToString())

    Dim document As XmlDocument
    Dim namespaceManager As XmlNamespaceManager
    namespaceManager = New XmlNamespaceManager(New NameTable())
    With namespaceManager
        .AddNamespace("asmv1", "urn:schemas-microsoft-com:asm.v1")
        .AddNamespace("asmv2", "urn:schemas-microsoft-com:asm.v2")
    End With

    document = New XmlDocument()
    document.Load(applicationManifestPath)

    Dim baseXPath As String
    baseXPath = "/asmv1:assembly/asmv2:dependency/asmv2:dependentOS/asmv2:osVersionInfo/asmv2:os"

    'Change minimum required OS Version.
    Dim node As XmlNode
    node = document.SelectSingleNode(baseXPath, namespaceManager)
    node.Attributes("majorVersion").Value = osVersion.Major.ToString()
    node.Attributes("minorVersion").Value = osVersion.Minor.ToString()
    node.Attributes("buildNumber").Value = osVersion.Build.ToString()
    node.Attributes("servicePackMajor").Value = osVersion.Revision.ToString()

    document.Save(applicationManifestPath)
End Sub
```

The command takes two arguments. The first argument is the path to the application manifest (that is, the folder in which the build process creates the manifest, typically `<ProjectName>.publish`). The second argument is the new operating system version.

5. On the **Build** menu, click **Build Solution**.
6. Copy the .exe file to a directory such as `C:\TEMP\ChangeOSVersionVB.exe`.

Next, invoke this command in a post-build event to change the application manifest.

To invoke a post-build event to change the application manifest

1. Create a Windows application for the project to be published. From the **File** menu, click **New**, and then

click **Project**.

2. In the **New Project** dialog box, in the **Visual Basic** node, select **Windows Desktop** and then the **Windows Forms App** template. Name the project `VBWinApp`.
3. With the project selected in **Solution Explorer**, on the **Project** menu, click **Properties**.
4. In the **Project Designer**, go to the **Publish** page and set **Publishing location** to `C:\TEMP`.
5. Publish the project by clicking **Publish Now**.

The manifest file will be built and put in `C:\TEMP\VBWinApp_1_0_0_0\VBWinApp.exe.manifest`. To view the manifest, right-click the file and click **Open with**, then click **Select the program from a list**, and then click **NotePad**.

Search in the file for the `<osVersionInfo>` element. For example, the version might be:

```
<os majorVersion="4" minorVersion="10" buildNumber="0" servicePackMajor="0" />
```

6. In the **Project Designer**, go to the **Compile** tab and click the **Build Events** button to open the **Build Events** dialog box.

7. In the **Post-build Event Command Line** box, enter the following command:

```
C:\TEMP\ChangeOSVersionVB.exe "$(TargetPath).manifest" 5.1.2600.0
```

When you build the project, this command will change the minimum operating system version in the application manifest to 5.1.2600.0.

The `$(TargetPath)` macro expresses the full path for the executable being created. Therefore, `$(TargetPath).manifest` will specify the application manifest created in the *bin* directory. Publishing will copy this manifest to the publishing location that you set earlier.

8. Publish the project again. Go to the **Publish** page and click **Publish Now**.

View the manifest again. To view the manifest, go to the publish directory, right-click the file and click **Open with** and then **Select the program from a list**, and then click **NotePad**.

The version should now read:

```
<os majorVersion="5" minorVersion="1" buildNumber="2600" servicePackMajor="0" />
```

See also

- [Compile page, Project Designer \(Visual Basic\)](#)
- [Publish page, Project Designer](#)
- [Pre-build event/Post-build event command line dialog box](#)
- [How to: Specify build events \(C#\)](#)

How to: Specify build events (C#)

10/18/2019 • 4 minutes to read • [Edit Online](#)

Use build events to specify commands that run before the build starts or after the build finishes. Build events execute only if the build successfully reaches those points in the build process.

When a project is built, pre-build events are added to a file named *PreBuildEvent.bat* and post-build events are added to a file named *PostBuildEvent.bat*. If you want to ensure error checking, add your own error-checking commands to the build steps.

Specify a build event

1. In **Solution Explorer**, select the project for which you want to specify the build event.
2. On the **Project** menu, click **Properties**.
3. Select the **Build Events** tab.
4. In the **Pre-build event command line** box, specify the syntax of the build event.

NOTE

Pre-build events do not run if the project is up to date and no build is triggered.

5. In the **Post-build event command line** box, specify the syntax of the build event.

NOTE

Add a `call` statement before all post-build commands that run *.bat* files. For example, `call C:\MyFile.bat` or `call C:\MyFile.bat call C:\MyFile2.bat`.

6. In the **Run the post-build event** box, specify under what conditions to run the post-build event.

NOTE

To add lengthy syntax, or to select any build macros from the [Pre-build event/post-build event command line dialog box](#), click the ellipsis button (...) to display an edit box.

The build event syntax can include any command that is valid at a command prompt or in a *.bat* file. The name of a batch file should be preceded by `call` to ensure that all subsequent commands are executed.

NOTE

If your pre-build or post-build event does not complete successfully, you can terminate the build by having your event action exit with a code other than zero (0), which indicates a successful action.

Example

The following procedure shows how to set the minimum operating system version in the application manifest by using an *.exe* command that is called from a post-build event (the *.exe.manifest* file in the project directory). The

minimum operating system version is a four-part number such as 4.10.0.0. To set the minimum operating system version, the command will change the `<dependentOS>` section of the manifest:

```
<dependentOS>
  <osVersionInfo>
    <os majorVersion="4" minorVersion="10" buildNumber="0" servicePackMajor="0" />
  </osVersionInfo>
</dependentOS>
```

Create an .exe command to change the application manifest

1. Create a new **Console App** project for the command. Name the project **ChangeOSVersionCS**.

2. In *Program.cs*, add the following line to the other `using` directives at the top of the file:

```
using System.Xml;
```

3. In the `ChangeOSVersionCS` namespace, replace the `Program` class implementation with the following code:

```

class Program
{
    /// <summary>
    /// This function sets the minimum operating system version for a ClickOnce application.
    /// </summary>
    /// <param name="args">
    /// Command Line Arguments:
    /// 0 - Path to application manifest (.exe.manifest)
    /// 1 - Version of OS
    /// </param>
    static void Main(string[] args)
    {
        string applicationManifestPath = args[0];
        Console.WriteLine("Application Manifest Path: " + applicationManifestPath);

        // Get version name.
        Version osVersion = null;
        if (args.Length >= 2 ){
            osVersion = new Version(args[1]);
        }else{
            throw new ArgumentException("OS Version not specified.");
        }
        Console.WriteLine("Desired OS Version: " + osVersion.ToString());

        XmlDocument document;
        XmlNamespaceManager namespaceManager;
        namespaceManager = new XmlNamespaceManager(new NameTable());
        namespaceManager.AddNamespace("asmv1", "urn:schemas-microsoft-com:asm.v1");
        namespaceManager.AddNamespace("asmv2", "urn:schemas-microsoft-com:asm.v2");

        document = new XmlDocument();
        document.Load(applicationManifestPath);

        string baseXPath;
        baseXPath = "/asmv1:assembly/asmv2:dependency/asmv2:dependentOS/asmv2:osVersionInfo/asmv2:os";

        // Change minimum required operating system version.
        XmlNode node;
        node = document.SelectSingleNode(baseXPath, namespaceManager);
        node.Attributes["majorVersion"].Value = osVersion.Major.ToString();
        node.Attributes["minorVersion"].Value = osVersion.Minor.ToString();
        node.Attributes["buildNumber"].Value = osVersion.Build.ToString();
        node.Attributes["servicePackMajor"].Value = osVersion.Revision.ToString();

        document.Save(applicationManifestPath);
    }
}

```

The command takes two arguments: the path of the application manifest (that is, the folder in which the build process creates the manifest, typically *Projectname.publish*), and the new operating system version.

4. Build the project.
5. Copy the .exe file to a directory such as C:\TEMP\ChangeOSVersionVB.exe.

Next, invoke this command in a post-build event to modify the application manifest.

Invoke a post-build event to modify the application manifest

1. Create a new **Windows Forms App** project and name it **CSWinApp**.
2. With the project selected in **Solution Explorer**, on the **Project** menu, choose **Properties**.
3. In the **Project Designer**, locate the **Publish** page and set **Publishing location** to C:\TEMP.
4. Publish the project by clicking **Publish Now**.

The manifest file is built and saved to `C:\TEMP\CSWinApp_1_0_0_0\CSWinApp.exe.manifest`. To view the manifest, right-click the file, click **Open with**, select **Select the program from a list**, and then click **Notepad**.

Search in the file for the `<osVersionInfo>` element. For example, the version might be:

```
<os majorVersion="4" minorVersion="10" buildNumber="0" servicePackMajor="0" />
```

5. Back in the **Project Designer**, click the **Build Events** tab and then click **Edit Post-build**.

6. In the **Post-build Event Command Line** box, enter the following command:

```
C:\TEMP\ChangeOSVersionCS.exe "$(TargetPath).manifest" 5.1.2600.0
```

When you build the project, this command changes the minimum operating system version in the application manifest to 5.1.2600.0.

Because the `$(TargetPath)` macro expresses the full path for the executable being created, `$(TargetPath).manifest` specifies the application manifest created in the *bin* directory. Publishing copies this manifest to the publishing location that you set earlier.

7. Publish the project again.

The manifest version should now read:

```
<os majorVersion="5" minorVersion="1" buildNumber="2600" servicePackMajor="0" />
```

See also

- [Build Events page, Project Designer \(C#\)](#)
- [Pre-build event/Post-build event command line dialog box](#)
- [How to: Specify build events \(Visual Basic\)](#)
- [Compile and build](#)

Configuring warnings in Visual Basic

10/18/2019 • 5 minutes to read • [Edit Online](#)

The Visual Basic compiler includes a set of warnings about code that may cause run-time errors. You can use that information to write cleaner, faster, better code with fewer bugs. For example, the compiler will produce a warning when the user attempts to invoke a member of an unassigned object variable, return from a function without setting the return value, or execute a `Try` block with errors in the logic to catch exceptions.

Sometimes the compiler provides extra logic on the user's behalf so that the user can focus on the task at hand, rather than on anticipating possible errors. In previous versions of Visual Basic, **Option Strict** was used to limit the additional logic that the Visual Basic compiler provides. Configuring warnings allows you to limit this logic in a more granular way, at the level of the individual warnings.

You may want to customize your project and turn off some warnings not pertinent to your application while turning other warnings into errors. This page explains how to turn individual warnings on and off.

Turning warnings off and on

There are two different ways to configure warnings: you can configure them using the **Project Designer**, or you can use the **/warnaserror** and **/nowarn** compiler options.

The **Compile** tab of the **Project Designer** page allows you to turn warnings on and off. Select the **Disable All Warnings** check box to disable all warnings; select the **Treat All Warnings as Errors** to treat all warnings as errors. Some individual warnings can be toggled as error or warning as desired in the displayed table.

When **Option Strict** is set to **Off**, **Option Strict** related warnings cannot be treated independently of each other. When **Option Strict** is set to **On**, the associated warnings are treated as errors, no matter what their status is.

When **Option Strict** is set to **Custom** by specifying `/optionstrict:custom` in the command line compiler, **Option Strict** warnings can be toggled on or off independently.

The **/warnaserror** command-line option of the compiler can also be used to specify whether warnings are treated as errors. You can add a comma delimited list to this option to specify which warnings should be treated as errors or warnings by using + or -. The following table details the possible options.

COMMAND-LINE OPTION	SPECIFIES
<code>/warnaserror+</code>	Treat all warnings as errors
<code>/warnaserror -</code>	Do not treat as warnings as errors. This is the default.
<code>/warnaserror+:<warning list></code>	Treat specific warnings as errors, listed by their error ID number in a comma delimited list r.
<code>/warnaserror-:<warning list></code>	Do not treat specific warnings as errors, listed by their error ID number in a comma delimited list.
<code>/nowarn</code>	Do not report warnings.
<code>/nowarn:<warning list></code>	Do not report specified warnings, listed by their error ID number in a comma delimited list.

The warning list contains the error ID numbers of the warnings that should be treated as errors, which can be used

with the command-line options to turn specific warnings on or off. If the warning list contains an invalid number, an error is reported.

Examples

This table of examples of command line arguments describes what each argument does.

ARGUMENT	DESCRIPTION
<code>vbc /warnaserror</code>	Specifies that all warnings should be treated as errors.
<code>vbc /warnaserror:42024</code>	Specifies that warning 42024 should be treated as an error.
<code>vbc /warnaserror:42024,42025</code>	Specifies that warnings 42024 and 42025 should be treated as errors.
<code>vbc /nowarn</code>	Specifies that no warnings should be reported.
<code>vbc /nowarn:42024</code>	Specifies that warning 42024 should not be reported.
<code>vbc /nowarn:42024,42025</code>	Specifies that warnings 42024 and 42025 should not be reported.

Types of warnings

Following is a list of warnings that you might want to treat as errors.

Implicit conversion warning

Generated for instances of implicit conversion. They do not include implicit conversions from an intrinsic numeric type to a string when using the `&` operator. Default for new projects is off.

ID: 42016

Late bound method invocation and overload resolution warning

Generated for instances of late binding. Default for new projects is off.

ID: 42017

Operands of type 'Object' warnings

Generated when operands of type `Object` occur that would create an error with **Option Strict On**. Default for new projects is on.

ID: 42018 and 42019

Declarations require 'As' clause warnings

Generated when a variable, function, or property declaration lacking an `As` clause would have created an error with **Option Strict On**. Variables that do not have a type assigned to them are assumed to be type `Object`. Default for new projects is on.

ID: 42020 (variable declaration), 42021 (function declaration), and 42022 (property declaration).

Possible null reference exception warnings

Generated when a variable is used before it has been assigned a value. Default for new projects is on.

ID: 42104, 42030

Unused local variable warning

Generated when a local variable is declared but never referred to. Default is on.

ID: 42024

Access of shared member through instance variable warning

Generated when accessing a shared member through an instance may have side effects, or when accessing a shared member through an instance variable is not the right-hand side of an expression or is being passed in as a parameter. Default for new projects is on.

ID: 42025

Recursive operator or property access warnings

Generated when the body of a routine uses the same operator or property it is defined in. Default for new projects is on.

ID: 42004 (operator), 42026 (property)

Function or operator without return value warning

Generated when the function or operator does not have a return value specified. This includes omitting a `Set` to the implicit local variable with the same name as the function. Default for new projects is on.

ID: 42105 (function), 42016 (operator)

Overloads modifier used in a module warning

Generated when `Overloads` is used in a `Module`. Default for new projects is on.

ID: 42028

Duplicate or overlapping catch blocks warnings

Generated when a `Catch` block is never reached due to its relation to other `Catch` blocks that have been defined. Default for new projects is on.

ID: 42029, 42031

See also

- [Error types](#)
- [Try...Catch...Finally statement](#)
- [/nowarn](#)
- [/warnaserror \(Visual Basic\)](#)
- [Compile page, Project Designer \(Visual Basic\)](#)
- [Compiler warnings that are off by default](#)

Walkthrough: Create a multiple-computer build environment

7/24/2019 • 11 minutes to read • [Edit Online](#)

You can create a build environment within your organization by installing Visual Studio on a host computer and then copying various files and settings to another computer so that it can participate in builds. You don't have to install Visual Studio on the other computer.

This document does not confer rights to redistribute the software externally or to provide build environments to third parties.

Disclaimer

This document is provided on a "as-is" basis. While we have tested the steps outlined, we are not able to exhaustively test every configuration. We will attempt to keep the document current with any additional information learned. Information and views expressed in this document, including URL and other Internet website references, may change without notice. Microsoft makes no warranties, express or implied, with respect to the information provided here. You bear the risk of using it.

This document does not provide you with any legal rights to any intellectual property in any Microsoft product. You may copy and use this document for your internal, reference purposes.

You have no obligation to give Microsoft any suggestions, comments or other feedback ("Feedback") relating to this document. However, any Feedback you voluntarily provide may be used in Microsoft Products and related specifications or other documentation (collectively, "Microsoft Offerings") which in turn may be relied upon by other third parties to develop their own products. Accordingly, if you do give Microsoft Feedback on any version of this document or the Microsoft Offerings to which they apply, you agree: (a) Microsoft may freely use, reproduce, license, distribute, and otherwise commercialize your Feedback in any Microsoft Offering; (b) You also grant third parties, without charge, only those patent rights necessary to enable other products to use or interface with any specific parts of a Microsoft Product that incorporate Your Feedback; and (c) You will not give Microsoft any Feedback (i) that you have reason to believe is subject to any patent, copyright or other intellectual property claim or right of any third party; or (ii) subject to license terms which seek to require any Microsoft Offering incorporating or derived from such Feedback, or other Microsoft intellectual property, to be licensed to or otherwise shared with any third party.

This walkthrough has been validated against the following operating systems:

- Windows 8 (x86 and x64)
- Windows 7 Ultimate
- Windows Server 2008 R2 Standard

After you complete the steps in this walkthrough, you can use the multiple-computer environment to build these kinds of apps:

- C++ desktop apps that use the Windows 8 SDK
- Visual Basic or C# desktop apps that target the .NET Framework 4.5

The multiple-computer environment can't be used to build these kinds of apps:

- UWP apps. To build UWP apps, you must install Visual Studio on the build computer.

- Desktop apps that target the .NET Framework 4 or earlier. To build these kinds of apps, you must install either Visual Studio or the .NET Reference Assemblies and Tools (from the Windows 7.1 SDK) on the build computer.

Prerequisites

Visual Studio with the **.NET desktop development** workload installed.

Install software on the computers

First, set up the host computer, and then set up the build computer.

By installing Visual Studio on the host computer, you create the files and settings that you will copy to the build computer later. You can install Visual Studio on an x86 or an x64 computer, but the architecture of the build computer must match the architecture of the host computer.

1. On the host computer, install Visual Studio.
2. On the build computer, install the .NET Framework 4.5 or later. To verify that it's installed, check that the **Version** entry in the registry subkey **HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\NET Framework Setup\NDP\v4\Full** has a value of **4.5** or higher.

Copy files from the host computer to the build computer

This section covers the copying of specific files, compilers, build tools, MSBuild assets, and registry settings from the host computer to the build computer. These instructions assume that you've installed Visual Studio in the default location on the host computer; if you installed in another location, adjust the steps accordingly.

- On an x86 computer, the default location is *C:\Program Files\Microsoft Visual Studio*
- On an x64 computer, the default location is *C:\Program Files (x86)\Microsoft Visual Studio*

Notice that the name of the *Program Files* folder depends on the operating system that's installed. On an x86 computer, the name is *Program Files*; on an x64 computer, the name is *Program Files (x86)*. Irrespective of the system architecture, this walkthrough refers to the *Program Files* folder as *%ProgramFiles%*.

NOTE

On the build computer, all of the relevant files must be on the same drive. However, the drive letter for that drive can be different than the drive letter for the drive where Visual Studio is installed on the host computer. In any case, you must account for the location of files when you create registry entries as described later in this document.

Copy the Windows SDK files to the build computer

1. If you have only the Windows SDK for Windows 8 installed, copy these folders recursively from the host computer to the build computer:
 - %ProgramFiles%\Windows Kits\8.0\bin\
 - %ProgramFiles%\Windows Kits\8.0\Catalogs\
 - %ProgramFiles%\Windows Kits\8.0\DesignTime\
 - %ProgramFiles%\Windows Kits\8.0\include\
 - %ProgramFiles%\Windows Kits\8.0\Lib\
 - %ProgramFiles%\Windows Kits\8.0\Redist\
 - %ProgramFiles%\Windows Kits\8.0\References\

If you also have these other Windows 8 kits...

- Microsoft Windows Assessment and Deployment Kit
- Microsoft Windows Driver Kit
- Microsoft Windows Hardware Certification Kit

...they might have installed files into the `%ProgramFiles%\Windows Kits\8.0` folders that are listed in the previous step, and their license terms might not allow build-server rights for those files. Check the license terms for every installed Windows kit to verify whether files may be copied to your build computer. If the license terms don't allow build-server rights, then remove the files from the build computer.

2. Copy the following folders recursively from the host computer to the build computer:

- `%ProgramFiles%\Microsoft SDKs\Windows\v8.0A\bin\NETFX 4.0 Tools\`
- `%ProgramFiles%\Common Files\Merge Modules\`
- `%ProgramFiles%\Microsoft Visual Studio\<version>\<edition>\VC\`
- `%ProgramFiles%\Microsoft Visual Studio\<version>\<edition>\Common7\Tools\ProjectComponents\`
- `%ProgramFiles%\MSBuild\Microsoft.Cpp\v4.0\v110\`
- `%ProgramFiles%\Reference Assemblies\Microsoft\Framework\.NETCore\v4.5\`
- `%ProgramFiles%\Reference Assemblies\Microsoft\Framework\.NETFramework\v4.5\`

3. Copy these files from the host computer to the build computer:

- `%ProgramFiles%\Microsoft Visual Studio\<version>\<edition>\Common7\IDE\msobj110.dll`
- `%ProgramFiles%\Microsoft Visual Studio\<version>\<edition>\Common7\IDE\mspdb110.dll`
- `%ProgramFiles%\Microsoft Visual Studio\<version>\<edition>\Common7\IDE\mspdbccore.dll`
- `%ProgramFiles%\Microsoft Visual Studio\<version>\<edition>\Common7\IDE\mspdbsrv.exe`
- `%ProgramFiles%\Microsoft Visual Studio\<version>\<edition>\Common7\IDE\msvcdis110.dll`
- `%ProgramFiles%\Microsoft Visual Studio\<version>\<edition>\Common7\Tools\makehm.exe`
- `%ProgramFiles%\Microsoft Visual Studio\<version>\<edition>\Common7\Tools\VCVarsQueryRegistry.bat`
- `%ProgramFiles%\Microsoft Visual Studio\<version>\<edition>\Common7\Tools\vsvars32.bat`

4. The following Visual C++ runtime libraries are required only if you run build outputs on the build computer—for example, as part of automated testing. The files are typically located in subfolders under the `%ProgramFiles%\Microsoft Visual Studio\<version>\<edition>\VC\redist\x86` or `%ProgramFiles%\Microsoft Visual Studio\<version>\<edition>\VC\redist\x64` folder, depending on the system architecture. On x86 systems, copy the x86 binaries to the `Windows\System32` folder. On x64 systems, copy the x86 binaries to the `Windows\SysWOW64` folder, and the x64 binaries to the `Windows\System32` folder.

- `\Microsoft.VC110.ATL\atl110.dll`
- `\Microsoft.VC110.CRT\msvcp110.dll`
- `\Microsoft.VC110.CRT\msvcr110.dll`

- \Microsoft.VC110.CXXAMP\vcamp110.dll
- \Microsoft.VC110.MFC\mfc110.dll
- \Microsoft.VC110.MFC\mfc110u.dll
- \Microsoft.VC110.MFC\mfcm110.dll
- \Microsoft.VC110.MFC\mfcm110u.dll
- \Microsoft.VC110.MFCLOC\mfc110chs.dll
- \Microsoft.VC110.MFCLOC\mfc110cht.dll
- \Microsoft.VC110.MFCLOC\mfc110deu.dll
- \Microsoft.VC110.MFCLOC\mfc110enu.dll
- \Microsoft.VC110.MFCLOC\mfc110esn.dll
- \Microsoft.VC110.MFCLOC\mfc110fra.dll
- \Microsoft.VC110.MFCLOC\mfc110ita.dll
- \Microsoft.VC110.MFCLOC\mfc110jpn.dll
- \Microsoft.VC110.MFCLOC\mfc110kor.dll
- \Microsoft.VC110.MFCLOC\mfc110rus.dll
- \Microsoft.VC110.OPENMP\vcomp110.dll

5. Copy only the following files from the *Debug_NonRedist\x86* or *Debug_NonRedist\x64* folder to the build computer, as described in [Prepare a test machine to run a debug executable](#). No other files may be copied.

- \Microsoft.VC110.DebugCRT\msvcp110d.dll
- \Microsoft.VC110.DebugCRT\msvcr110d.dll
- \Microsoft.VC110.DebugCXXAMP\vcamp110d.dll
- \Microsoft.VC110.DebugMFC\mfc110d.dll
- \Microsoft.VC110.DebugMFC\mfc110ud.dll
- \Microsoft.VC110.DebugMFC\mfcm110d.dll
- \Microsoft.VC110.DebugMFC\mfcm110ud.dll
- \Microsoft.VC110.DebugOpenMP\vcomp110d.dll

Create registry settings

You must create registry entries to configure settings for MSBuild.

1. Identify the parent folder for registry entries. All of the registry entries are created under the same parent key. On an x86 computer, the parent key is **HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft**. On an x64 computer, the parent key is **HKEY_LOCAL_MACHINE\SOFTWARE\Wow6432Node\Microsoft**. Irrespective of the system architecture, this walkthrough refers to the parent key as %RegistryRoot%.

NOTE

If the architecture of your host computer differs from that of your build computer, make sure to use the appropriate parent key on each computer. This is especially important if you're automating the export process.

Also, if you're using a different drive letter on the build computer than the one that you're using on the host computer, make sure to change the values of the registry entries to match.

2. Create the following registry entries on the build computer. All of these entries are strings (Type == "REG_SZ" in the registry). Set the values of these entries the same as the values of the comparable entries on the host computer.

- **%RegistryRoot%\NETFramework\v4.0.30319\AssemblyFoldersEx\VCMSBuild Public Assemblies@(Default)**
- **%RegistryRoot%\Microsoft SDKs\Windows\v8.0@InstallationFolder**
- **%RegistryRoot%\Microsoft SDKs\Windows\v8.0A@InstallationFolder**
- **%RegistryRoot%\Microsoft SDKs\Windows\v8.0A\WinSDK-NetFx40Tools@InstallationFolder**
- **%RegistryRoot%\Microsoft SDKs\Windows\v8.0A\WinSDK-NetFx40Tools-x86@InstallationFolder**
- **%RegistryRoot%\VisualStudio\11.0@Source Directories**
- **%RegistryRoot%\VisualStudio\11.0\Setup\VC@ProductDir**
- **%RegistryRoot%\VisualStudio\SxS\VC7@FrameworkDir32**
- **%RegistryRoot%\VisualStudio\SxS\VC7@FrameworkDir64**
- **%RegistryRoot%\VisualStudio\SxS\VC7@FrameworkVer32**
- **%RegistryRoot%\VisualStudio\SxS\VC7@FrameworkVer64**
- **%RegistryRoot%\VisualStudio\SxS\VC7@11.0**
- **%RegistryRoot%\VisualStudio\SxS\VS7@11.0**
- **%RegistryRoot%\Windows Kits\Installed Roots@KitsRoot**
- **%RegistryRoot%\MSBuild\ToolsVersions\4.0\11.0@VCTargetsPath**
- **%RegistryRoot%\MSBuild\ToolsVersions\4.0\11.0@VCTargetsPath10**
- **%RegistryRoot%\MSBuild\ToolsVersions\4.0\11.0@VCTargetsPath11**

On an x64 build computer, also create the following registry entry and refer to the host computer to determine how to set it.

- **%RegistryRoot%\Microsoft SDKs\Windows\v8.0A\WinSDK-NetFx40Tools-x64@InstallationFolder**

If your build computer is x64 and you want to use the 64-bit version of MSBuild, or if you're using Team Foundation Server Build Service on an x64 computer, create the following registry entries in the native 64-bit registry. Refer to the host computer to determine how to set these entries.

- **HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\VisualStudio\11.0\Setup\VS@ProductDir**
- **HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\MSBuild\ToolsVersions\4.0\11.0@VCTarg**

etsPath

- **HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\MSBuild\ToolsVersions\4.0\11.0@VCTargetsPath10**
- **HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\MSBuild\ToolsVersions\4.0\11.0@VCTargetsPath11**

Set environment variables on the build computer

To use MSBuild on the build computer, you must set the PATH environment variables. You can use `vcvarsall.bat` to set the variables, or you can manually configure them.

Use `vcvarsall.bat` to set environment variables

Open a **Command Prompt** window on the build computer and run `%Program Files%\Microsoft Visual Studio\<version>\<edition>\VC\vcvarsall.bat`. You can use a command-line argument to specify the toolset you want to use—x86, native x64, or x64 cross-compiler. If you don't specify a command-line argument, the x86 toolset is used.

This table describes the supported arguments for `vcvarsall.bat`:

VCVARSALL.BAT ARGUMENT	COMPILER	BUILD COMPUTER ARCHITECTURE	BUILD OUTPUT ARCHITECTURE
x86 (default)	32-bit Native	x86, x64	x86
x86_amd64	x64 Cross	x86, x64	x64
amd64	x64 Native	x64	x64

If `vcvarsall.bat` runs successfully—that is, no error message is displayed—you can skip the next step and continue at the [Install MSBuild assemblies to the Global Assembly Cache \(GAC\) on the build computer](#) section of this document.

Manually set environment variables

1. To manually configure the command-line environment, add this path to the PATH environment variable:
 - `%Program Files%\Microsoft Visual Studio\<version>\<edition>\Common7\IDE`
2. Optionally, you can also add the following paths to the PATH variable to make it easier to use MSBuild to build your solutions.

If you want to use the native 32-bit MSBuild, add these paths to the PATH variable:

- `%Program Files%\Microsoft SDKs\Windows\v8.0A\bin\NETFX 4.0 Tools`
- `%windir%\Microsoft.NET\Framework\v4.0.30319`

If you want to use the native 64-bit MSBuild, add these paths to the PATH variable:

- `%Program Files%\Microsoft SDKs\Windows\v8.0A\bin\NETFX 4.0 Tools\x64`
- `%windir%\Microsoft.NET\Framework64\v4.0.30319`

Install MSBuild assemblies to the Global Assembly Cache (GAC) on the build computer

MSBuild requires some additional assemblies to be installed to the GAC on the build computer.

1. Copy the following assemblies from the host computer to the build computer. Because they will be installed

to the GAC, it doesn't matter where you put them on the build computer.

- %ProgramFiles%\MSBuild\Microsoft.Cpp\v4.0\v110\Microsoft.Build.CPPTasks.Common.v110.dll
 - %ProgramFiles%\Microsoft Visual Studio\<version>\<edition>\Common7\IDE\CommonExtensions\Microsoft\VC\Project\Microsoft.VisualStudio.Project.VisualC.VCProjectEngine.dll
 - %ProgramFiles%\Microsoft Visual Studio\<version>\<edition>\Common7\IDE\PublicAssemblies\Microsoft.VisualStudio.VCProjectEngine.dll
2. To install the assemblies to the GAC, locate *gacutil.exe* on the build computer—typically, it's in %ProgramFiles%\Microsoft SDKs\Windows\v8.0A\bin\NETFX 4.0 Tools\. If you can't locate this folder, repeat the steps in the [Copy files from the host computer to the build computer](#) section of this walkthrough.

Open a **Command Prompt** window that has administrative rights and run this command for each file:

```
gacutil -i <file>
```

NOTE

A reboot may be required for an assembly to fully install into the GAC.

Build projects

You can use Azure Pipelines to build Visual Studio projects and solutions, or you can build them on the command line. When you use Azure Pipelines to build projects, it invokes the MSBuild executable that corresponds to the system architecture. On the command line, you can use either 32-bit MSBuild or 64-bit MSBuild, and you can choose the architecture of MSBuild by setting the PATH environment variable or by directly invoking the architecture-specific MSBuild executable.

To use *msbuild.exe* at the command prompt, run the following command, in which *solution.sln* is a placeholder for the name of your solution.

```
msbuild solution.sln
```

For more information about how to use MSBuild on the command line, see [Command-line reference](#).

Create the build environment so that it can be checked into source control

You can create a build environment that can be deployed to various computers and doesn't require "GAC"-ing files or modifying registry settings. The following steps are just one way to accomplish this. Adapt these steps to the unique characteristics of your build environment.

NOTE

You must disable incremental building so that *tracker.exe* will not throw an error during a build. To disable incremental building, set this build parameter:

```
msbuild solution.sln /p:TrackFileAccess=false
```

1. Create a *Depot* directory on the host computer.

These steps refer to the directory as %Depot%.

2. Copy the directories and files as described in the [Copy files from the host computer to the build computer](#)

section of this walkthrough, except paste them under the %Depot% directory that you just created. For example, copy from %ProgramFiles%\Windows Kits\8.0\bin to %Depot%\Windows Kits\8.0\bin.

3. When the files are pasted in %Depot%, make these changes:

- In %Depot%\MSBuild\Microsoft.Cpp\v4.0\v110\Microsoft.CPP.Targets, \Microsoft.Cpp.InvalidPlatforms.targets\, \Microsoft.cppbuild.targets\, and \Microsoft.CppCommon.targets\, change every instance of

```
AssemblyName="Microsoft.Build.CppTasks.Common.v110, Version=4.0.0.0, Culture=neutral,  
PublicKeyToken=b03f5f7f11d50a3a"
```

to

```
AssemblyFile="$(VCTargetsPath11)Microsoft.Build.CppTasks.Common.v110.dll".
```

The former naming relies on the assembly being GAC'ed.

- In %Depot% \MSBuild\Microsoft.Cpp\v4.0\v110\Microsoft.CPPClean.Targets, change every instance of

```
AssemblyName="Microsoft.Build.CppTasks.Common.v110, Version=4.0.0.0, Culture=neutral,  
PublicKeyToken=b03f5f7f11d50a3a"
```

to

```
AssemblyFile="$(VCTargetsPath11)Microsoft.Build.CppTasks.Common.v110.dll".
```

4. Create a *.props* file—for example, *Partner.AutoImports.props*—and put it at the root of the folder that contains your projects. This file is used to set variables that are used by MSBuild to find various resources. If the variables are not set by this file, they are set by other *.props* files and *.targets* files that rely on registry values. Because we aren't setting any registry values, these variables would be empty and the build would fail. Instead, add this to *Partner.AutoImports.props*:

```
<?xml version="1.0" encoding="utf-8"?>  
<!-- This file must be imported by all project files at the top of the project file. -->  
<Project ToolsVersion="4.0"  
xmlns="http://schemas.microsoft.com/developer/msbuild/2003">  
<PropertyGroup  
<VCTargetsPath>$(&#39;$(DepotRoot)MSBuild\Microsoft.Cpp\v4.0\v110\&#39;</VCTargetsPath>  
<VCTargetsPath11>$(&#39;$(DepotRoot)MSBuild\Microsoft.Cpp\v4.0\v110\&#39;</VCTargetsPath11>  
<MSBuildExtensionsPath>$(&#39;$(DepotRoot)MSBuild</MSBuildExtensionsPath>  
<MSBuildExtensionsPath32>$(&#39;$(DepotRoot)MSBuild</MSBuildExtensionsPath32>  
<VCInstallDir_110>$(&#39;$(DepotRoot)Microsoft Visual Studio\2017\Enterprise\VC\</VCInstallDir_110>  
<VCInstallDir>$(&#39;$(VCInstallDir_110)</VCInstallDir>  
<WindowsKitRoot>$(&#39;$(DepotRoot)Windows Kits\</WindowsKitRoot>  
<WindowsSDK80Path>$(&#39;$(WindowsKitRoot)</WindowsSDK80Path>  
<WindowsSdkDir_80>$(&#39;$(WindowsKitRoot)8.0\</WindowsSdkDir_80>  
<WindowsSdkDir>$(&#39;$(WindowsSDKDir_80)</WindowsSdkDir>  
<WindowsSdkDir_80A>$(&#39;$(DepotRoot)Microsoft SDKs\Windows\v8.0A\</WindowsSdkDir_80A>  
</PropertyGroup  
</Project>
```

5. In each of your project files, add the following line at the top, after the `<Project Default Targets...>` line.

```
<Import Project="$(MSBuild)::GetDirectoryNameOfFileAbove($(MSBuildThisFileDirectory),  
Partner.AutoImports.props)\Partner.AutoImports.props"/>
```

6. Change the command-line environment as follows:

- Set Depot=*location of the Depot directory that you created in step 1*

- Set path=%path%;*location of MSBuild on the computer*;%Depot%\Windows\System32;%Depot%\Windows\SysWOW64;%Depot%\Microsoft Visual Studio 15.0\Common7\IDE\

For native 64-bit building, point to the 64-bit version of MSBuild.

6. Change the command-line environment as follows:

- Set Depot=*location of the Depot directory that you created in step 1*
- Set path=%path%;*location of MSBuild on the computer*;%Depot%\Windows\System32;%Depot%\Windows\SysWOW64;%Depot%\Microsoft Visual Studio 16.0\Common7\IDE\

For native 64-bit building, point to the 64-bit version of MSBuild.

See also

- [Prepare a test machine to run a debug executable](#)
- [Command-line reference](#)

Testing tools in Visual Studio

10/28/2019 • 2 minutes to read • [Edit Online](#)

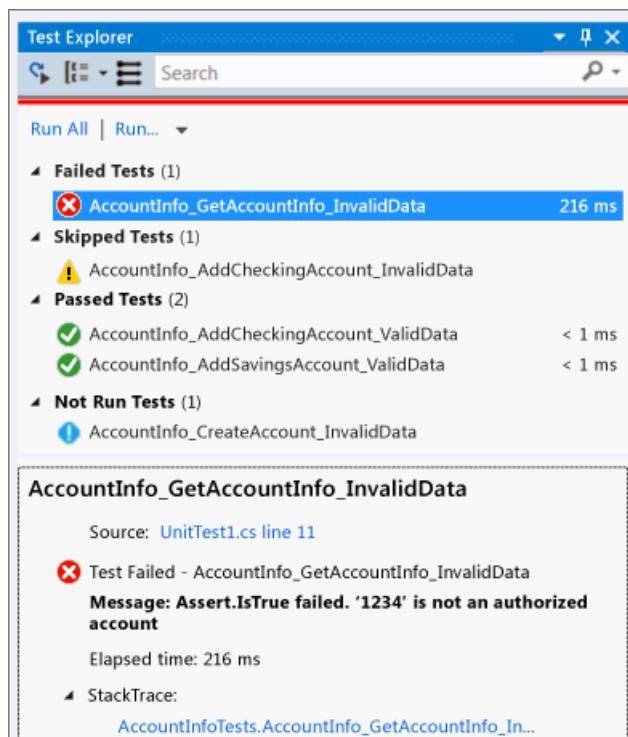
Visual Studio testing tools can help you and your team develop and sustain high standards of code excellence.

NOTE

Unit testing is available in all editions of Visual Studio. Other testing tools, such as Live Unit Testing, IntelliTest, and Coded UI Test, are only available in Visual Studio Enterprise edition. For more information about editions see [Compare Visual Studio IDEs](#).

Test Explorer

The **Test Explorer** window helps developers create, manage, and run unit tests. You can use the Microsoft unit test framework or one of several third-party and open source frameworks.



Test Explorer

17 6 5 6

Test	Duration	Error Message
>UserSentimentAnalysis.Tests (17)	83 ms	
>UserSentimentAnalysis.Tests (17)	83 ms	
EmojiTests (5)	6 ms	
EmojiClothingTest	5 ms	
EmojiExtraSpecialCharatersTest	< 1 ms	
EmojiFaceSearchTest	1 ms	
EmojiHeartsTest	< 1 ms	
TearsOfJoyTest	< 1 ms	
HomeControllerTests (6)		
TwitterDataModelTests (6)	77 ms	
ActiveCognitiveServiceTest	74 ms	Assert.AreEqual failed. Expected...
AverageTweetTest	3 ms	Test method UserSentimentAna...
GetTweetSentimentTest	< 1 ms	Assert.AreEqual failed. Expected...
HistoricalTweetTest	< 1 ms	Assert.AreEqual failed. Expected...
MultipleAverageTweetTest	< 1 ms	
ParseTweetTest	< 1 ms	Assert.AreEqual failed. Expected...

Group Summary

TwitterDataModelTests

Tests in group: 6

Total Duration: 77 ms

Outcomes

1 Passed
5 Failed

The screenshot shows the Visual Studio Test Explorer interface. At the top, there are navigation icons and a status bar with '17' (tests), '6' (passed), '5' (failed), and '6' (pending). Below this is a table with columns for 'Test', 'Duration', and 'Error Message'. The table lists various test cases, including groups like 'UserSentimentAnalysis.Tests' and 'HomeControllerTests', and specific test methods like 'EmojiClothingTest' and 'ParseTweetTest'. Most tests have a duration of less than 1 ms, except for one at 77 ms. The 'TwitterDataModelTests' group is expanded, showing six failing tests due to 'Assert.AreEqual' failures. At the bottom, there's a 'Group Summary' section for 'TwitterDataModelTests' with a breakdown of outcomes: 1 Passed and 5 Failed.

- [Get started with unit testing](#)
- [Run unit tests with Test Explorer](#)
- [Test Explorer FAQ](#)
- [Install third-party unit test frameworks](#)

Visual Studio is also extensible and opens the door for third-party unit testing adapters such as NUnit and xUnit.net. In addition, the code clone capability goes hand-in-hand with delivering high-quality software by helping you identify blocks of semantically similar code that may be candidates for common bug fixes or refactoring.

 ChutzpahDemo_20120705.5 - Build succeeded

[View Summary](#) | [View Log](#) - [Open Drop Folder](#) | [Diagnostics](#) ▾ | [<No Quality Assigned>](#) ▾ | [Actions](#) ▾

 Mathew Aniyan triggered ChutzpahDemo (UnitTest) for changeset 249
Ran for 87 seconds (Hosted Build Controller), completed 4 days ago

Latest Activity

Build last modified by Elastic Build (mathewan) 4 days ago.

Request Summary

[Request 55](#), requested by Mathew Aniyan 4 days ago, Completed

Summary

Debug | Any CPU

- ▷ 0 error(s), 2 warning(s)
- ▷ \$/UnitTest/Sources/ChutzpahDemo/ChutzpahDemo.sln compiled
- ◀ 1 test run completed - 100% pass rate
[buildguest@WIN-1CQH2N24PSU 2012-07-05 10:06:44_Any CPU_Debug](#), 3 of 3 test(s) passed

No Code Coverage Results

Test Results

buildguest@WIN-1CQH2N24PSU				Run	Debug	Group By:	[None]
 Test run passed Results: 3/3 passed; Item(s) checked: 0							
Result	Test Name	ID	Error Message				
 Passed	hello test	hello test::C:\a\Binaries\tests.hello test					
 Passed	a basic test	a basic test::C:\a\Binaries\tests.a basic test					
 Passed	add 5	add 5::C:\a\Binaries\tests.add 5					

Live Unit Testing

[Live Unit Testing](#) automatically runs unit tests in the background, and graphically displays code coverage and test results in the Visual Studio code editor.

IntelliTest

IntelliTest automatically generates unit tests and test data for your managed code. IntelliTest improves coverage and dramatically reduces the effort to create and maintain unit tests for new or existing code.

IntelliTest Exploration Results - stopped

TaxCalculator.CalculateExemption

5 ✅ 2 ❌ 10/10 blocks, 0/0 asserts, 13 runs

	target	employee	ic	result(target)
✖ 1	new TaxCalcu	null		null
✅ 2	new TaxCalcu	new Employee(An	new HRAexe	new TaxCal
✅ 3	new TaxCalcu	new Employee(An	new HRAexe	new TaxCal
✅ 4	new TaxCalcu	new Employee(An	new HRAexe	new TaxCal
✅ 5	new TaxCalcu	new Employee(An	new HRAexe	new TaxCal
✖ 6	new TaxCalcu	new Employee(An	new HRAexe	
✅ 7	new TaxCalcu	new Employee(An	new HRAexe	new TaxCal

Details

```

[TestMethod]
[PexGeneratedBy(typeof(TaxCalculator))]
[PexRaisedException(typeof(NullReferenceException))]
public void CalculateExemptionForRentThresholdException453()
{
    uint u;
    TaxCalculator s0 = new TaxCalculator();
    u = this.CalculateExemptionForRentThresholdException453();
}

```

- Generate unit tests for your code with IntelliTest

- [IntelliTest – One test to rule them all](#)
- [IntelliTest reference manual](#)

Code coverage

[Code coverage](#) determines what proportion of your project's code is actually being tested by coded tests such as unit tests. To guard effectively against bugs, your tests should exercise or "cover" a large proportion of your code.

Code coverage analysis can be applied to both managed and unmanaged (native) code.

Code coverage is an option when you run test methods using Test Explorer. The results table shows the percentage of the code that was run in each assembly, class, and method. In addition, the source editor shows you which code has been tested.

- [Use code coverage to determine how much code is being tested](#)
- [Unit testing, code coverage and code clone analysis with Visual Studio \(Lab\)](#)
- [Customize code coverage analysis](#)

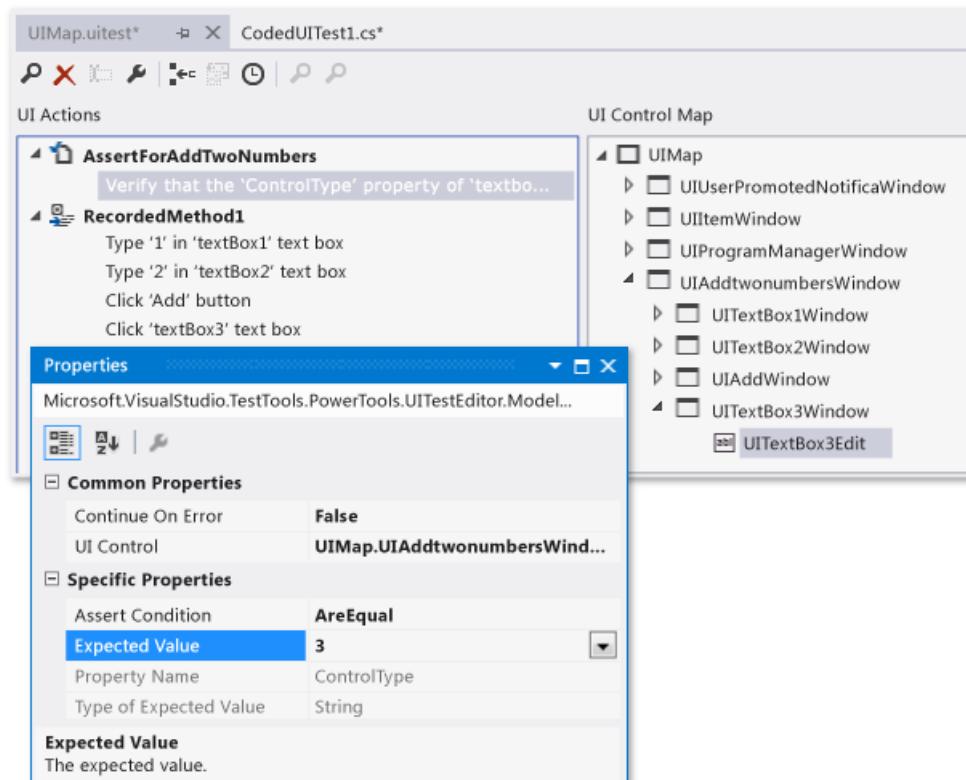
Microsoft Fakes

[Microsoft Fakes](#) help you isolate the code you're testing by replacing other parts of the application with stubs or shims.

User interface testing with Coded UI and Selenium

Coded UI tests provide a way to create fully automated tests to validate the functionality and behavior of your application's user interface. They can automate UI testing across a variety of technologies, including XAML-based UWP apps, browser apps, and SharePoint apps.

Whether you choose best-of-breed Coded UI Tests or generic browser-based UI testing with Selenium, Visual Studio provides all the tools you need.



- [Use UI automation to test your code](#)
- [Get started creating, editing, and maintaining a coded UI test](#)

- [Test UWP apps with coded UI tests](#)
- [Introduction to coded UI tests with Visual Studio Enterprise \(Lab\)](#)

Load testing

[Load testing](#) simulates load on a server application by running unit tests and web performance tests.

Related scenarios

- [Exploratory & manual testing \(Azure Test Plans\)](#)
- [Load testing \(Azure Test Plans\)](#)
- [Continuous testing \(Azure Test Plans\)](#)
- [Code analysis tools](#)

Visual Studio SDK

11/4/2019 • 2 minutes to read • [Edit Online](#)

The Visual Studio SDK helps you extend Visual Studio features or integrate new features into Visual Studio. You can distribute your extensions to other users, as well as to the Visual Studio Marketplace. The following are some of the ways in which you can extend Visual Studio:

- Add commands, buttons, menus, and other UI elements to the IDE
- Add tool windows for new functionality
- Extend IntelliSense for a given language, or provide IntelliSense for new programming languages
- Use light bulbs to provide hints and suggestions that help developers write better code
- Enable support for a new language
- Add a custom project type
- Reach millions of developers via the Visual Studio Marketplace

If you've never written a Visual Studio extension before, you should find more information about these features and at [Starting to develop Visual Studio extensions](#).

Install the Visual Studio SDK

The Visual Studio SDK is an optional feature in Visual Studio setup. You can also install the VS SDK later on. For more information, see [Install the Visual Studio SDK](#).

What's new in the Visual Studio 2017 SDK

The Visual Studio SDK has some new features such as the VSIX v3 format as well as breaking changes, which may require you to update your extension. For more information, see [What's new in the Visual Studio 2017 SDK](#).

Visual Studio user experience guidelines

Get great tips for designing the UI for your extension in [Visual Studio user experience guidelines](#).

You can also learn how to make your extension look great on high DPI devices with the [Address DPI issues](#) article.

Take advantage of the [Image service and catalog](#) for great image management and support for high DPI and theming.

Find and install existing Visual Studio extensions

You can find Visual Studio extensions in the **Extensions and Updates** dialog on the **Tools** menu. For more information, see [Find and Use Visual Studio Extensions](#). You can also find extensions in the [Visual Studio Marketplace](#)

Visual Studio SDK reference

You can find the Visual Studio SDK API reference at [Visual Studio SDK Reference](#).

Visual Studio SDK samples

You can find open source examples of VS SDK extensions on GitHub at [Visual Studio Samples](#). This GitHub repo contains samples that illustrate various extensible features in Visual Studio.

Other Visual Studio SDK resources

If you have questions about the VSSDK or want to share your experiences developing extensions, you can use the [Visual Studio Extensibility Forum](#) or the [ExtendVS Gitter Chatroom](#).

You can find more information in the [VSX Arcana blog](#) and a number of blogs written by Microsoft MVPs:

- [Favorite Visual Studio extensions](#)
- [Visual Studio extensibility](#)
- [Extending Visual Studio](#)

See also

- [Create an extension with a menu command](#)
- [How to: Migrate extensibility projects to Visual Studio 2017](#)
- [FAQ: Converting add-ins to VS Package extensions](#)
- [Manage multiple threads in managed code](#)
- [Extend menus and commands](#)
- [Add commands to toolbars](#)
- [Extend and customizing tool windows](#)
- [Editor and language service extensions](#)
- [Extend projects](#)
- [Extend user settings and options](#)
- [Create custom project and item templates](#)
- [Extend properties and the property window](#)
- [Extend other parts of Visual Studio](#)
- [Use and providing services](#)
- [Manage VS Packages](#)
- [Visual Studio isolated shell](#)
- [Ship Visual Studio extensions](#)
- [Inside the Visual Studio SDK](#)
- [Support for the Visual Studio SDK](#)
- [Visual Studio SDK reference](#)

Analyze and model your architecture

10/18/2019 • 2 minutes to read • [Edit Online](#)

Make sure your app meets architectural requirements by using Visual Studio architecture and modeling tools to design and model your app.

- Understand existing program code more easily by using Visual Studio to visualize the code's structure, behavior, and relationships.
- Educate your team in the need for respecting architectural dependencies.
- Create models at different levels of detail throughout the application lifecycle as part of your development process.

See [Scenario: Change your design using visualization and modeling](#).

Article reference

Visualize code: - See the code's organization and relationships by creating code maps. Visualize dependencies between assemblies, namespaces, classes, methods, and so on. - See the class structure and members for a specific project by creating class diagrams from code. - Find conflicts between your code and its design by creating dependency diagrams to validate code.	- Visualize code - Working with Classes and Other Types (Class Designer) - Video: Understand design from code with Visual Studio 2015 code maps - Video: Validate your architecture dependencies in real time
Define the architecture: - Define and enforce constraints on dependencies between the components of your code by creating dependency diagrams.	- Video: Validate architecture dependencies with Visual Studio (Channel 9)
Validate your system with the requirements and intended design: - Validate code dependencies with dependency diagrams that describe the intended architecture and prevent changes that might conflict with the design.	- Video: Validate architecture dependencies with Visual Studio (Channel 9)
Customize models and diagrams: - Create your own domain-specific languages.	- Modeling SDK for Visual Studio - Domain-Specific Languages
Generate text using T4 templates: - Use text blocks and control logic inside templates to generate text-based files. - T4 template build with MSBuild included in Visual Studio	- Code Generation and T4 Text Templates

<p>Share models, diagrams, and code maps using Team Foundation version control:</p>	
--	--

- Put code maps, projects, and dependency diagrams under Team Foundation version control so you can share them.

To see which editions of Visual Studio support each feature, see [Edition support for architecture and modeling tools](#)

Types of models and typical uses

Code maps

Code maps help you see the organization and relationships in your code.

Typical uses:

- Examine program code so you can better understand its structure and its dependencies, how to update it, and estimate the cost of proposed changes.

See:

- [Map dependencies across your solutions](#)
- [Use code maps to debug your applications](#)
- [Find potential problems using code map analyzers](#)

Dependency diagrams

Dependency diagrams let you define the structure of an application as a set of layers or blocks with explicit dependencies. Live validation shows conflicts between dependencies in the code and dependencies described on a dependency diagram.

Typical uses:

- Stabilize the structure of the application through numerous changes over its life.
- Discover unintentional dependency conflicts before checking in changes to the code.

See:

- [Create dependency diagrams from your code](#)
- [Dependency Diagrams: Reference](#)
- [Validate code with dependency diagrams](#)

Domain-specific language (DSL)

A DSL is a notation that you design for a specific purpose. In Visual Studio, it's usually graphical.

Typical uses:

- Generate or configure parts of the application. Work is required to develop the notation and tools. The result can be a better fit to your domain than a UML customization.
- For large projects or in product lines where the investment in developing the DSL and its tools is returned by its use in more than one project.

See:

- [Modeling SDK for Visual Studio - Domain-Specific Languages](#)

See also

- [What's new for modeling in Visual Studio 2017](#)
- [DevOps and Application Lifecycle Management](#)

Personalize the Visual Studio IDE

10/18/2019 • 2 minutes to read • [Edit Online](#)

You can personalize Visual Studio in various ways to best support your own development style and requirements. Many of your settings roam with you across Visual Studio instances—see [Synchronized settings](#). This article briefly describes different personalizations and where you can find more information.

NOTE

This topic applies to Visual Studio on Windows. For Visual Studio for Mac, see [Customize the Visual Studio for Mac IDE](#).

Default settings

You can choose a default collection of settings that optimizes Visual Studio for your type of development. For more information, see [Environment settings](#).

General environment options

Many personalization options are exposed through the [Environment Options](#) dialog box. There are two ways to access this dialog box:

- On the menu bar, choose **Tools** > **Options**, and if it's not already expanded, expand the **Environment** node.
- Press **Ctrl+Q**, type **environment** in the search box, and then choose **Environment** > **General** from the results.

TIP

When the Options dialog box appears, you can press **F1** for help on the various settings on that page.

Environment color themes

To change the color theme between light, dark and blue, type **environment** in the search box, and then choose **Environment** > **General**. In the **Options** dialog box, change the **Color theme** option.

To change colorization options in the editor, type **environment** in the search box, and then choose **Environment** > **Fonts and Colors**. See [How to: Change fonts and colors](#).

Main menu casing

You can change the main menu casing between **Title Case** ("File") and **All Caps** ("FILE"). Type **environment** in the search box, select **Environment** > **General**, and then change the **Apply title case styling to menu bar** option.

Customize menus and toolbars

To add or remove menu or toolbar items, see [How to: Customize menus and toolbars](#).

Start page

To create a custom start page for you and your team, see [Customize the Start page](#).

Window layouts

You can define and save multiple window layouts and switch between them. For example, you can define one layout for coding and one for debugging. To arrange window positions and behavior and save custom layouts, see [Customize window layouts](#).

External tools

You can customize the **Tools** menu to launch external tools. For more information, see [Manage external tools](#).

See also

- [Environment settings](#)
- [Visual Studio IDE overview](#)
- [Quickstart: First look at the Visual Studio IDE](#)
- [Customize the Visual Studio for Mac IDE](#)

Personalize the Visual Studio IDE and Editor

10/18/2019 • 2 minutes to read • [Edit Online](#)

In this 5-10 minute tutorial, we'll customize the Visual Studio color theme by selecting the dark theme. We'll also customize the colors for two different types of text in the text editor.

If you haven't already installed Visual Studio, go to the [Visual Studio downloads](#) page to install it for free.

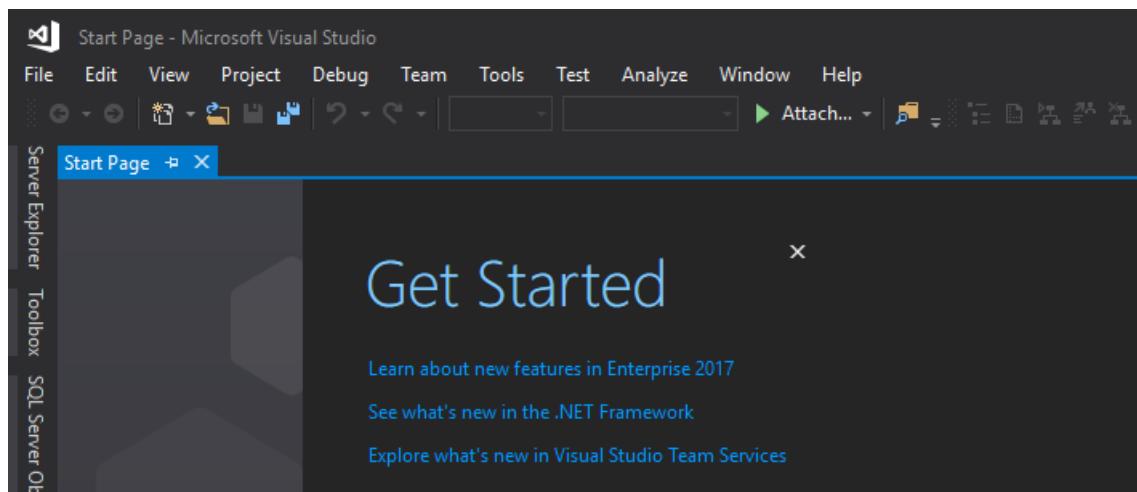
If you haven't already installed Visual Studio, go to the [Visual Studio downloads](#) page to install it for free.

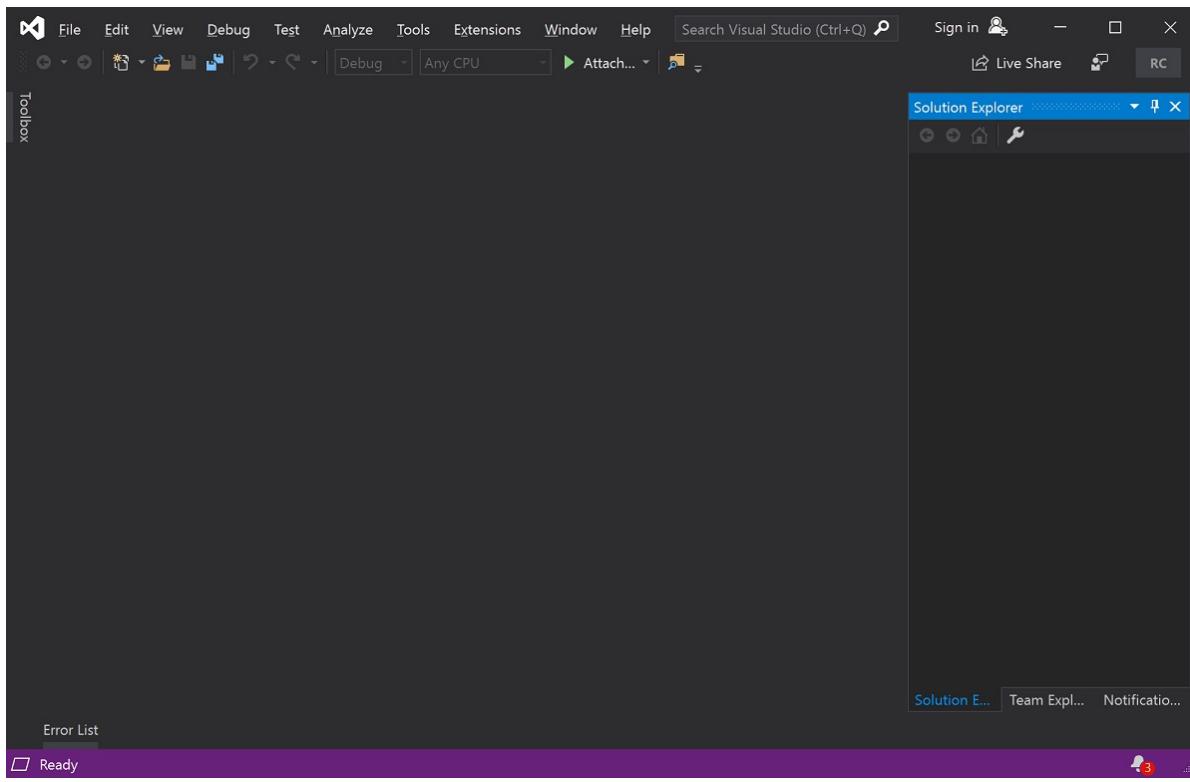
Set the color theme

The default color theme for Visual Studio's user interface is called **Blue**. Let's change it to **Dark**.

1. On the menu bar, which is the row of menus such as **File** and **Edit**, choose **Tools > Options**.
2. On the **Environment > General** options page, change the **Color theme** selection to **Dark**, and then choose **OK**.

The color theme for the entire Visual Studio development environment (IDE) changes to **Dark**.





TIP

You can install additional predefined themes by installing the [Visual Studio Color Theme Editor](#) from the [Visual Studio Marketplace](#). After you install this tool, additional color themes appear in the **Color theme** drop-down list.

Change text color

Now we'll customize some text colors for the editor. First, let's create a new XML file to see the default colors.

1. From the menu bar, choose **File** > **New** > **File**.
2. In the **New File** dialog box, under the **General** category, choose **XML File**, and then choose **Open**.
3. Paste the following XML below the line that contains `<?xml version="1.0" encoding="utf-8"?>`.

```

<Catalog>
  <Book id="bk101">
    <Author>Garghentini, Davide</Author>
    <Title>XML Developer's Guide</Title>
    <Genre>Computer</Genre>
    <Price>44.95</Price>
    <PublishDate>2000-10-01</PublishDate>
    <Description>
      An in-depth look at creating applications with XML.
    </Description>
  </Book>
  <Book id="bk102">
    <Author>Garcia, Debra</Author>
    <Title>Midnight Rain</Title>
    <Genre>Fantasy</Genre>
    <Price>5.95</Price>
    <PublishDate>2000-12-16</PublishDate>
    <Description>
      A former architect battles corporate zombies, an evil
      sorceress, and her own childhood to become queen of the world.
    </Description>
  </Book>
</Catalog>

```

Notice that the line numbers are a turquoise-blue color, and the XML attributes (such as `id="bk101"`) are a light blue color. We're going to change the text color for these items.

```

<?xml version="1.0" encoding="utf-8"?>
<Catalog>
  <Book id="bk101">
    <Author>Garghentini, Davide</Author>
    <Title>XML Developer's Guide</Title>
    <Genre>Computer</Genre>
    <Price>44.95</Price>
    <PublishDate>2000-10-01</PublishDate>
    <Description>
      An in-depth look at creating applications with XML.
    </Description>
  </Book>
  <Book id="bk102">
    <Author>Garcia, Debra</Author>
    <Title>Midnight Rain</Title>
    <Genre>Fantasy</Genre>
    <Price>5.95</Price>
    <PublishDate>2000-12-16</PublishDate>
    <Description>
      A former architect battles corporate zombies, an evil
      sorceress, and her own childhood to become queen of the world.
    </Description>
  </Book>
</Catalog>

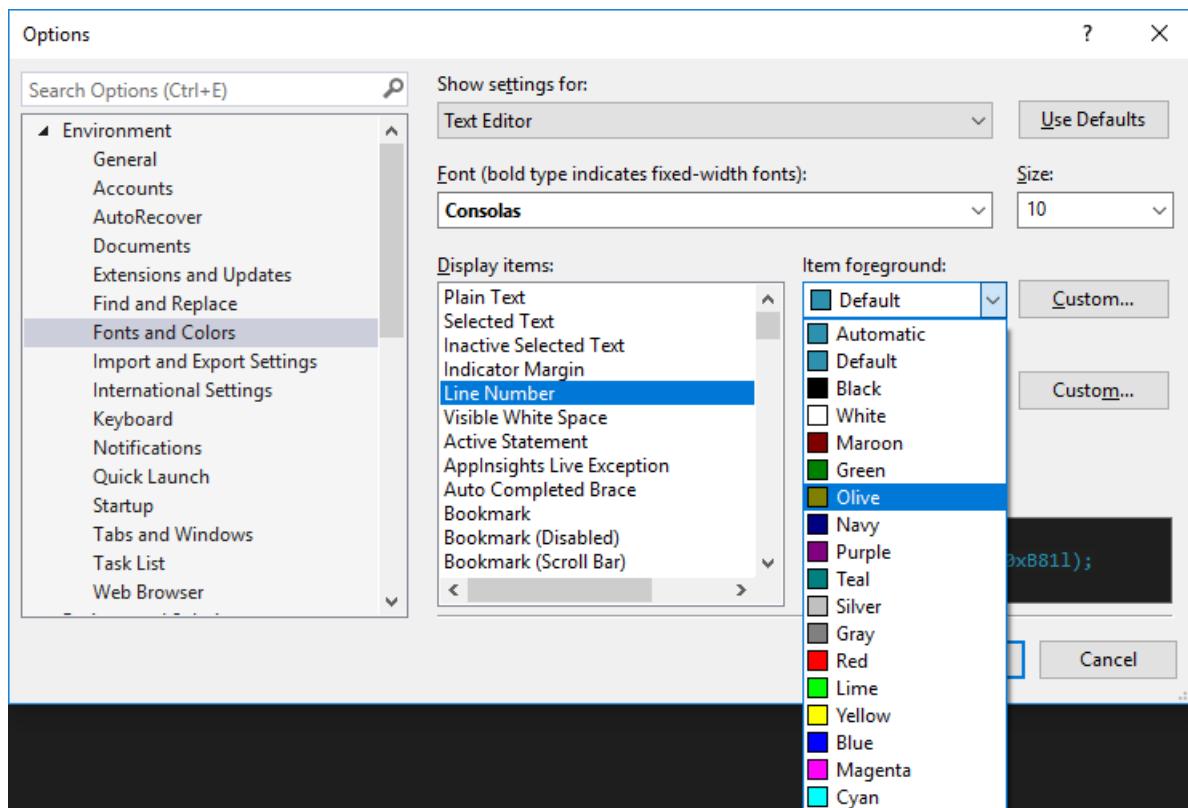
```

4. To open the **Options** dialog box, choose **Tools > Options** from the menu bar.

5. Under **Environment**, choose the **FONT AND COLORS** category.

Notice that the text under **Show settings for** says **Text Editor**—this is what we want. Expand the drop-down list just to see the extensive list of places where you can customize fonts and text color.

6. To change the color of the line numbers text, in the **DISPLAY ITEMS** list, choose **LINE NUMBER**. In the **ITEM FOREGROUND** box, choose **Olive**.



Some languages have their own specific fonts and colors settings. If you are a C++ developer and you want to change the color used for functions, for example, you can look for **C++ Functions** in the **Display items** list.

7. Before we exit out of the dialog box, let's also change the color of XML attributes. In the **Display items** list, scroll down to **XML Attribute** and select it. In the **Item foreground** box, choose **Lime**. Choose **OK** to save our selections and close the dialog box.

The line numbers are now an olive color, and the XML attributes are a bright, lime green. If you open another file type, such as a C++ or C# code file, you'll see that the line numbers also appear in the olive color.

```
<?xml version="1.0" encoding="utf-8"?>
<Catalog>
  <Book id="bk101">
    <Author>Garghentini, Davide</Author>
    <Title>XML Developer's Guide</Title>
    <Genre>Computer</Genre>
    <Price>44.95</Price>
    <PublishDate>2000-10-01</PublishDate>
    <Description>
      An in-depth look at creating applications with XML.
    </Description>
  </Book>
  <Book id="bk102">
    <Author>Garcia, Debra</Author>
    <Title>Midnight Rain</Title>
    <Genre>Fantasy</Genre>
    <Price>5.95</Price>
    <PublishDate>2000-12-16</PublishDate>
    <Description>
      A former architect battles corporate zombies, an evil
      sorceress, and her own childhood to become queen of the world.
    </Description>
  </Book>
</Catalog>
```

We explored just a couple ways of customizing the colors in Visual Studio. We hope that you'll explore the other customization options in the **Options** dialog box, to truly make Visual Studio your own.

See also

- [Customize the editor](#)
- [Visual Studio IDE Overview](#)

Environment settings for Visual Studio

10/18/2019 • 2 minutes to read • [Edit Online](#)

When you open Visual Studio for the first time, you can optimize the development environment for the type of development that you do the most by choosing a collection of settings. Each collection optimizes elements such as keyboard shortcuts, window layouts, project and item templates, and command visibility.

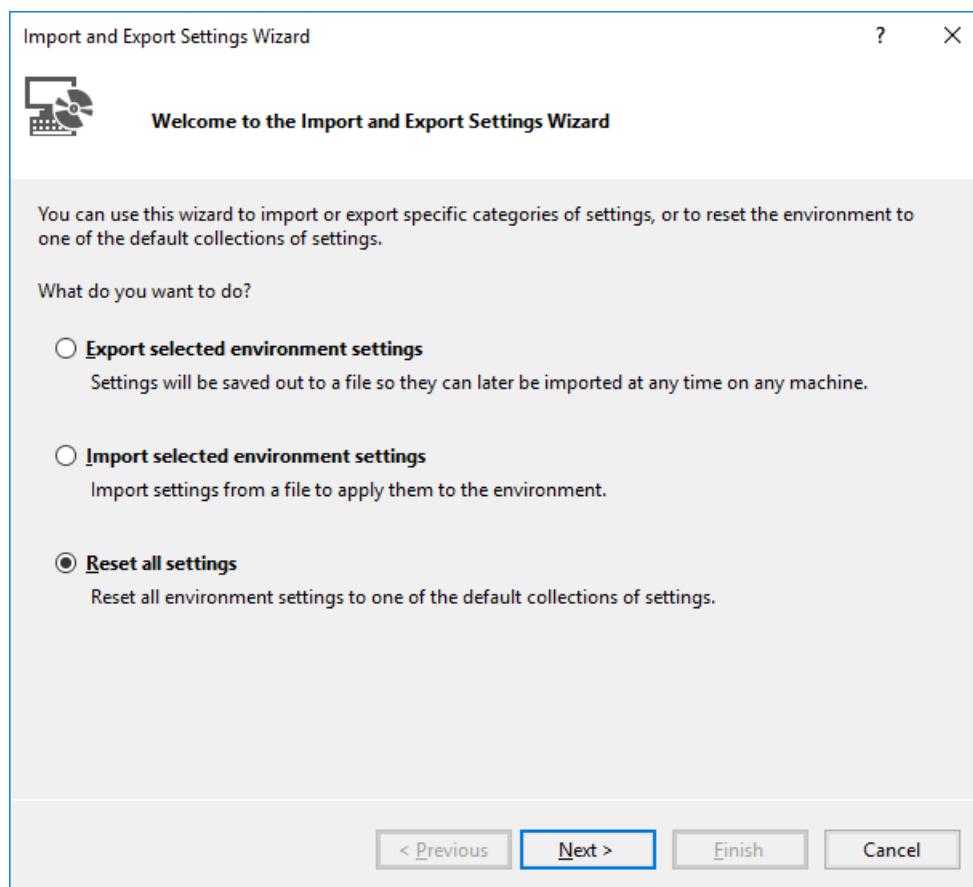
The following settings collections are available:

- General
- JavaScript
- Visual Basic
- Visual C#
- Visual C++
- Web Development
- Web Development (Code Only)

Reset settings

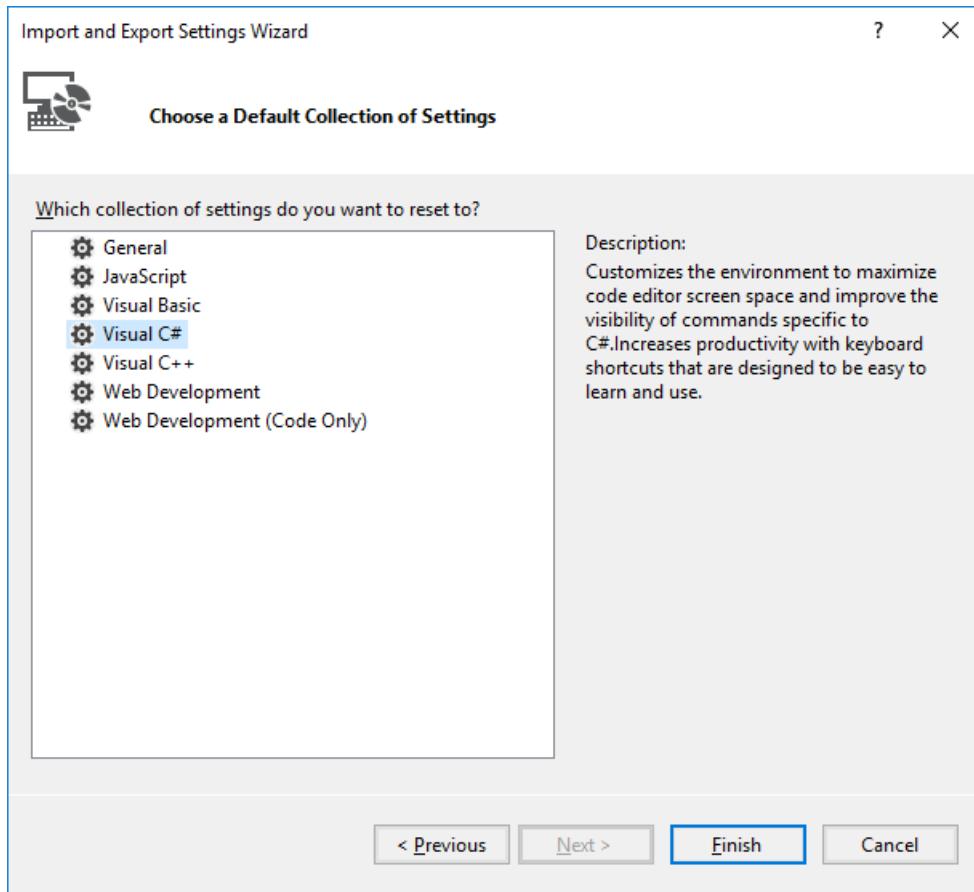
To change your development settings after you open Visual Studio for the first time, follow these steps:

1. Select **Tools > Import and Export Settings** from the menu bar to open the **Import and Export Settings Wizard**.
2. In the **Import and Export Settings Wizard**, select **Reset all settings**, and then select **Next**.



3. On the **Save Current Settings** page, select either **Yes** or **No**, and then select **Next**.

4. On the **Choose a Default Collection of Settings** page, choose a collection, and then select **Finish**.



5. On the **Reset Complete** page, select **Close**.

See also

- [Synchronize settings across multiple computers](#)
- [Personalize the Visual Studio IDE](#)

Synchronize Visual Studio settings across multiple computers

10/18/2019 • 3 minutes to read • [Edit Online](#)

When you sign in to Visual Studio on multiple computers using the same personalization account, your settings can be synchronized across the computers.

Synchronized settings

By default, the following settings are synchronized:

- Development settings. You select a collection of settings the first time you open Visual Studio, but you can change the selection anytime. For more information, see [Environment settings](#).
- User-defined command aliases. For more information about how to define command aliases, see [Visual Studio command aliases](#).
- User-defined window layouts in **Window > Manage Window Layouts** page.
- The following options in the **Tools > Options** pages:
 - Theme and menu bar casing settings on the **Environment > General** options page.
 - All settings on the **Environment > Fonts and Colors** options page.
 - All keyboard shortcuts on the **Environment > Keyboard** options page.
 - All settings on the **Environment > Tabs and Windows** options page.
 - All settings on the **Environment > StartUp** options page.
 - All settings on the **Text Editor** options pages, for example, [code style preferences](#).
 - All settings on the **XAML Designer** options pages.

Turn off synchronized settings on a particular computer

Synchronized settings for Visual Studio are turned on by default. You can turn off synchronized settings on a computer by going to the **Tools > Options > Environment > Accounts** page and unchecking **Synchronize settings across devices when signed into Visual Studio**.

As an example, if you decide not to synchronize Visual Studio's settings on computer "A", any setting changes made on computer "A" do not appear on computer "B" or computer "C". Computers "B" and "C" will continue to synchronize with each other, but not with computer "A".

NOTE

If you choose not to synchronize settings by deselecting the option on the **Tools > Options > Environment > Accounts** page, other versions or editions of Visual Studio that you have on the same computer aren't affected. Those side-by-side installations of Visual Studio will continue to synchronize their settings (unless you uncheck the option there, too).

Synchronize settings across Visual Studio family products and editions

Settings are synchronized across versions and editions of Visual Studio installed *side-by-side*. Settings are also synchronized across Visual Studio family products, including Blend for Visual Studio. However, an individual family product may have its own settings that aren't shared with Visual Studio. For example, settings specific to Blend for Visual Studio on computer "A" are not shared with Visual Studio on computers "A" or "B".

Side-by-side synchronized settings

Certain settings like tool window layout aren't shared between different side-by-side installations of Visual Studio. The *CurrentSettings.vssettings* file in `%userprofile%\Documents\Visual Studio 2017\Settings` is in an installation-specific folder that is similar to `%localappdata%\Microsoft\VisualStudio\15.0_xxxxxxx\Settings`.

NOTE

To use the new installation-specific settings, do a fresh installation. When you upgrade an existing Visual Studio installation, it uses the existing shared location.

If you currently have side-by-side installations of Visual Studio and want to use the new installation-specific settings file location, follow these steps:

1. Upgrade to Visual Studio 2017 version 15.3 or later.
2. Use the **Import and Export Settings Wizard** to export all your existing settings to some location outside of the `%localappdata%\Microsoft\VisualStudio\15.0_xxxxxxx` folder.
3. Open the **Developer Command Prompt for VS 2017** and run `devenv /resetuserdata`.
4. Open Visual Studio and import the saved settings from the exported settings file.

Certain settings like tool window layout aren't shared between different side-by-side installations of Visual Studio. The *CurrentSettings.vssettings* file in `%userprofile%\Documents\Visual Studio 2019\Settings` is in an installation-specific folder that is similar to `%localappdata%\Microsoft\VisualStudio\16.0_xxxxxxx\Settings`.

Reset synchronized settings

To reset all settings to their defaults, sign in to Visual Studio, and then select **Tools > Import and Export Settings** to open the **Import and Export Settings Wizard**. Select **Reset all settings** and then follow the remaining steps of the wizard.

See also

- [Personalize the IDE](#)
- [Environment settings](#)
- [Environment > Accounts Options dialog box](#)

How to: Change fonts and colors in Visual Studio

10/18/2019 • 2 minutes to read • [Edit Online](#)

You can customize the color of the IDE frame and tool windows in Visual Studio in several ways.

TIP

For information about how to change the colors of the code editor, see [How to: Change fonts and colors in the editor](#).

Change the color theme of the IDE

1. On the menu bar, choose **Tools** > **Options**.
2. In the options list, choose **Environment** > **General**.
3. In the **Color theme** list, choose either the default **Blue** theme, **Dark**, or **Light**.

NOTE

When you change a color theme, text in the IDE reverts to the default or previously customized fonts and sizes.

TIP

You can create and edit Visual Studio themes by installing the [Visual Studio Color Theme Editor](#).

Use Windows high contrast colors

Choose the **Left Alt+Left Shift+PrtScn** keys.

WARNING

This option sets high contrast for all applications and UI on the current computer.

Change IDE fonts

You can change the font and text size for all windows and dialog boxes in the IDE. You can choose to customize only certain windows and other text elements.

To change the font and size of all text in the IDE

1. On the menu bar, choose **Tools** > **Options**.
2. In the options list, choose **Environment** > **Fonts and Colors**.
3. In the **Show settings for** list, choose **Environment Font**.

TIP

If you want to change the font for tool windows only, in the **Show settings for** list, choose **All Text Tool Windows**.

4. In the **Font** list, choose a font.
5. In the **Size** list, choose a text size, and then choose the **OK** button.

See also

- [Accessibility features of Visual Studio](#)
- [How to: Change fonts and colors in the editor](#)

How to: Customize menus and toolbars in Visual Studio

10/18/2019 • 2 minutes to read • [Edit Online](#)

You can customize Visual Studio not only by adding and removing toolbars and menus on the menu bar, but also by adding and removing commands on any given toolbar or menu.

WARNING

After you customize a toolbar or menu, make sure that its check box remains selected in the **Customize** dialog box. Otherwise, your changes won't persist after you close and reopen Visual Studio.

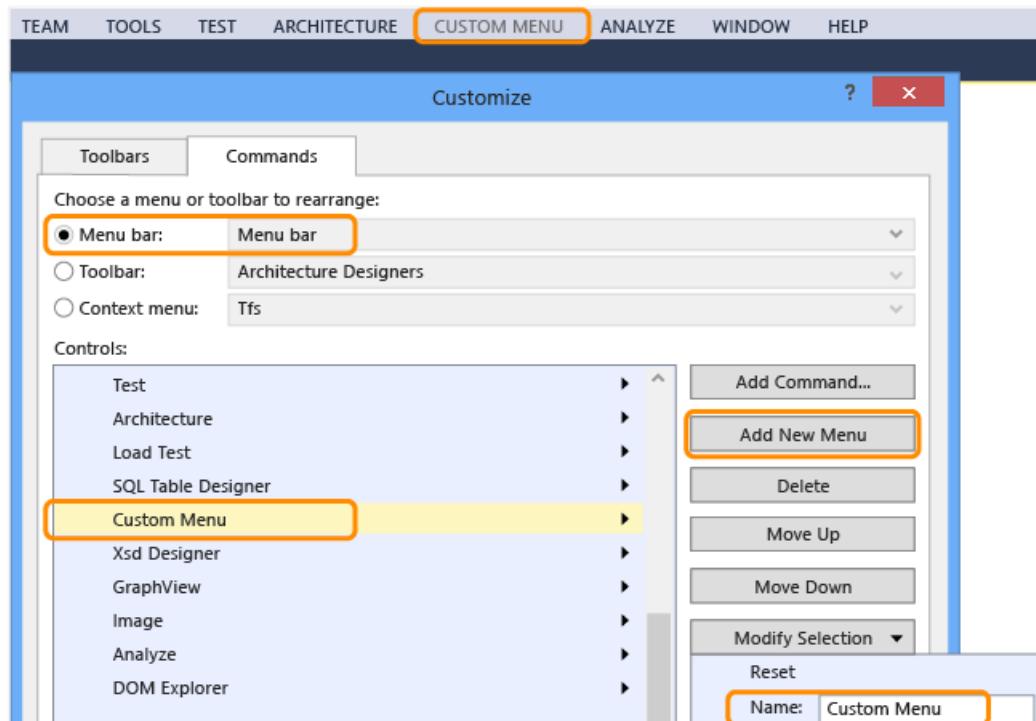
Add, remove, or move a menu on the menu bar

1. On the menu bar, choose **Tools** > **Customize**.

The **Customize** dialog box opens.

2. On the **Commands** tab, leave the **Menu bar** option button selected, leave **Menu Bar** selected in the list next to that option, and then perform one of the following sets of steps:

- To add a menu, choose the **Add New Menu** button, choose the **Modify Selection** button, and then name the menu that you want to add.



- To remove a menu, choose it in the **Controls** list, and then choose the **Delete** button.
- To move a menu within the menu bar, choose the menu in the **Controls** list, and then choose the **Move Up** or **Move Down** button.

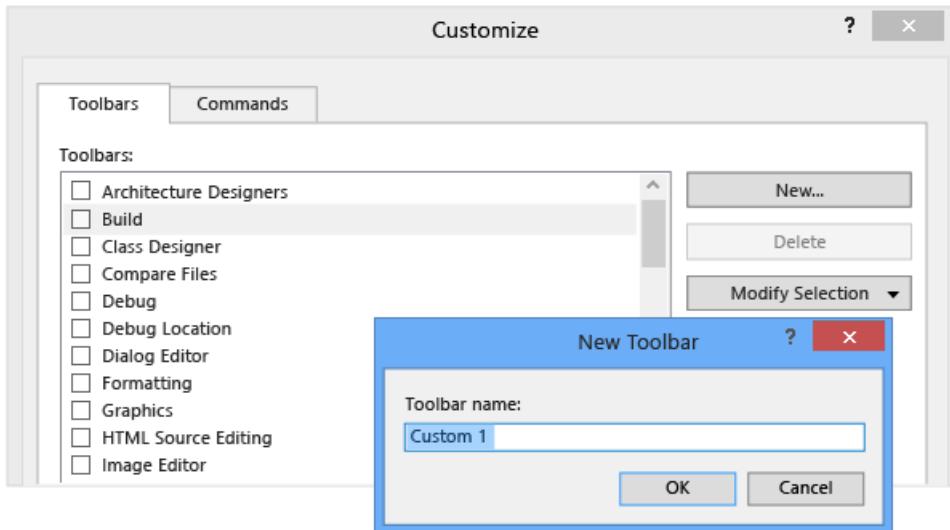
Add, remove, or move a toolbar

1. On the menu bar, choose **Tools** > **Customize**.

The **Customize** dialog box opens.

2. On the **Toolbar** tab, perform one of the following sets of steps:

- To add a toolbar, choose the **New** button, specify a name for the toolbar that you want to add, and then choose the **OK** button.



- To remove a custom toolbar, choose it in the **Toolbars** list, and then choose the **Delete** button.

IMPORTANT

You can delete toolbars that you create but not default toolbars.

- To move a toolbar to a different docking location, choose it in the **Toolbars** list, choose the **Modify Selection** button, and then choose a location in the list that appears.

You can also drag a toolbar by its left edge to move it anywhere in the main docking area.

NOTE

For more information about how to improve the usability and accessibility of toolbars, see [How to: Set IDE accessibility options](#).

Customize a menu or a toolbar

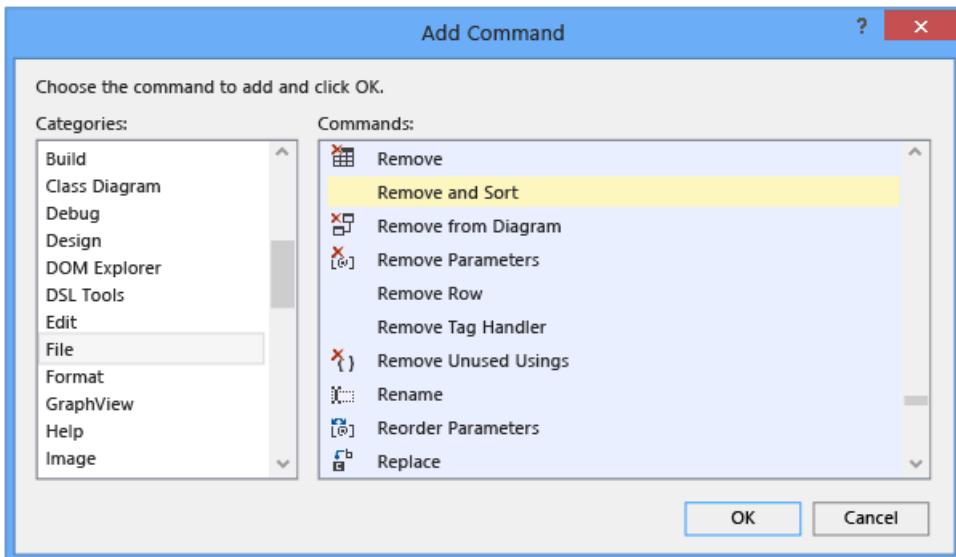
1. On the menu bar, choose **Tools** > **Customize**.

The **Customize** dialog box opens.

2. On the **Commands** tab, choose the option button for the type of element that you want to customize.
3. In the list for that type of element, choose the menu or toolbar that you want to customize, and then perform one of the following sets of steps:

- To add a command, choose the **Add Command** button.

In the **Add Command** dialog box, choose an item in the **Categories** list, choose an item in the **Commands** list, and then choose the **OK** button.



- To delete a command, choose it in the **Controls** list, and then choose the **Delete** button.
- To reorder commands, choose a command in the **Controls** list, and then choose the **Move Up** or **Move Down** button.
- To group commands under a horizontal line, choose the first command in the **Controls** list, choose the **Modify Selection** button, and then choose **Begin a Group** in the menu that appears.

Reset a menu or a toolbar

1. On the menu bar, choose **Tools > Customize**.

The **Customize** dialog box opens.

2. On the **Commands** tab, choose the option button for the type of element that you want to reset.
3. In the list for that type of element, choose the menu or toolbar that you want to reset.
4. Choose the **Modify Selection** button, and then choose **Reset** in the menu that appears.

You can also reset all menus and toolbars by choosing the **Reset All** button.

See also

- [Personalize the IDE](#)
- [Customize the editor](#)

Customize window layouts in Visual Studio

10/18/2019 • 9 minutes to read • [Edit Online](#)

In Visual Studio, you can customize the position, size, and behavior of windows to create window layouts that work best for various development workflows. When you customize the layout, the IDE remembers it. For example, if you change the docking location of **Solution Explorer** and then close Visual Studio, the next time that you open Visual Studio, even if you're working on another computer, **Solution Explorer** will be docked in that same location.

You can also name and save a custom layout and then switch between layouts with a single command. For example, you could create a layout for editing and a layout for debugging, and switch between them by using the **Window > Apply Window Layout** menu command.

Kinds of windows

Tool and document windows

The IDE has two basic window types, *tool windows* and *document windows*. Tool windows include **Solution Explorer**, **Server Explorer**, **Output Window**, **Error List**, the designers, the debugger windows, and so on. Document windows contain source code files, arbitrary text files, config files, and so on. Tool windows can be resized and dragged by their title bar. Document windows can be dragged by their tab. Right-click on the tab or title bar to set other options on the window.

The **Window** menu shows options for docking, floating and hiding windows in the IDE. Right click on a window tab or title bar to see additional options for that specific window. You can display more than one instance of certain tool windows at a time. For example, you can display more than one web browser window, and you can create additional instances of some tool windows by choosing **New Window** on the **Window** menu.

Preview tab (document windows)

In the **Preview** tab, you can view files in the editor without opening them. You can preview files by choosing them in **Solution Explorer**, during debugging when you step into files, with **Go to Definition**, and when you browse through results of a search. Preview files appear in a tab on the right side of the document tab well. The file opens for editing if you modify it or choose **Open**.

Tab groups

Tab groups extend your ability to manage limited workspace while you are working with two or more open documents in the IDE. You can organize multiple document windows and tool windows into either vertical or horizontal tab groups and shuffle documents from one tab group to another.

Split windows

When you have to view or edit two locations at once in a document, you can split windows. To divide your document into two independently scrolling sections, click **Split** on the **Window** menu. Click **Remove Split** on the **Window** menu to restore the single view.

Toolbars

Toolbars can be arranged by dragging, or by using the **Customize** dialog box. For more information about how to position and customize toolbars, see [How to: Customize menus and toolbars](#).

Arrange and dock windows

A document window or tool window can be *docked*, so that it has a position and size within the IDE window

frame, or floating as a separate window independent of the IDE. Tool windows can be docked anywhere inside the IDE frame; some tool windows can be docked as tabbed windows in the editor frame. Document windows can be docked within the editor frame, and they can be pinned to their current position in the tab order. You can dock multiple windows to float together in a *raft* over or outside of the IDE. Tool windows can also be hidden or minimized.

You can arrange windows in the following ways:

- Pin document windows to the left of the tab well.
- Tab-dock windows to the editing frame.
- Dock tool windows to the edge of a frame in the IDE.
- Float document or tool windows over or outside the IDE.
- Hide tool windows along the edge of the IDE.
- Display windows on different monitors.
- Reset window placement to the default layout or to a saved custom layout.

Arrange tool and document windows by dragging, using commands on the **Window** menu, or by right-clicking the title bar of the window to be arranged.

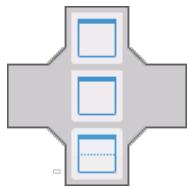
Dock windows

When you click and drag the title bar of a tool window, or the tab of document window, a guide diamond appears. During the drag operation, when the mouse cursor is over one of the arrows in the diamond, a shaded area will appear that shows you where the window will be docked if you release the mouse button now.

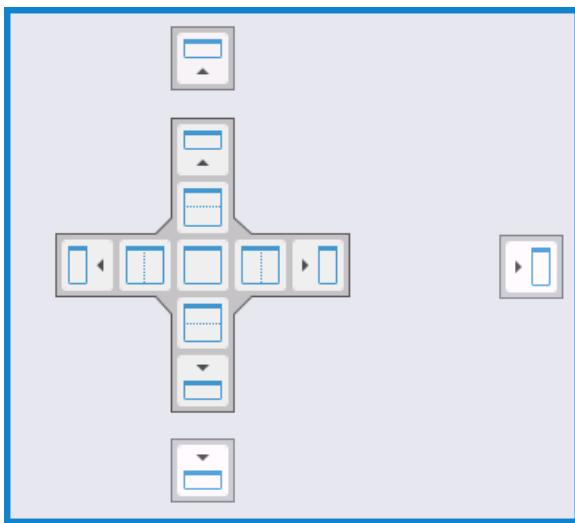
To move a dockable window without snapping it into place, press the **Ctrl** key while you drag the window.

To return a tool window or document window to its most recent docked location, press **Ctrl** while you double-click the title bar or tab of the window.

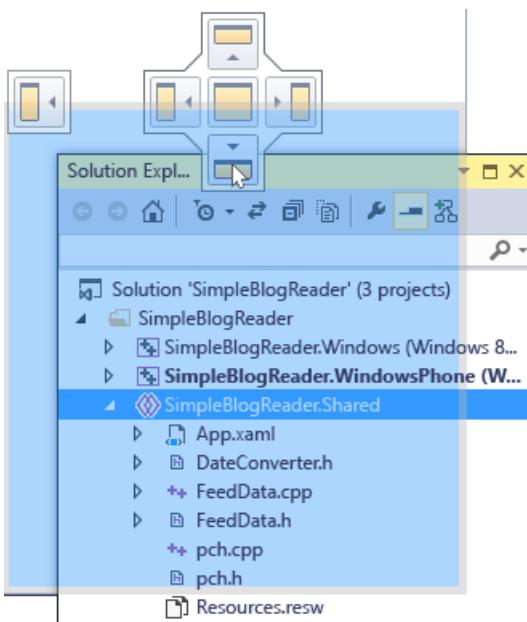
The following illustration shows the guide diamond for document windows, which can only be docked within the editing frame:



Tool windows can be fastened to one side of a frame in the IDE or within the editing frame. A guide diamond appears when you drag a tool window to another location to help you to easily redock the window.

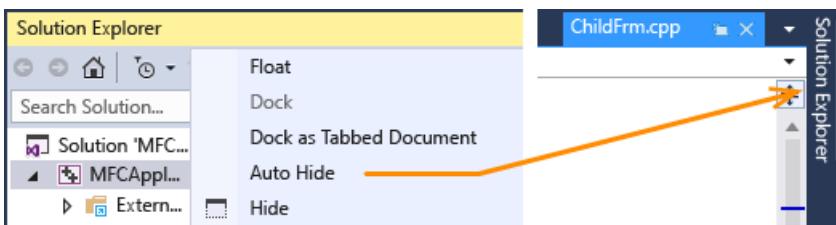


The following illustration shows **Solution Explorer** being docked in a new location that's demarcated by the blue shaded area:



Close and auto-hide tool windows

You can close a tool window by clicking the **X** in the upper right of the title bar. To reopen the window, use its keyboard shortcut or menu command. Tool windows support a feature named *auto hide*, which causes a window to slide out of the way when you use a different window. When a window is autohidden, its name appears on a tab at the edge of the IDE. To use the window again, point to the tab so that the window slides back into view.



NOTE

To set whether auto hide operates on tool windows individually or as docked groups, select or clear **Auto Hide button affects active tool windows only** in the **Options** dialog box. For more information, see [General, Environment, Options dialog box](#).

NOTE

Tool windows that have auto hide enabled may temporarily slide into view when the window has focus. To hide the window again, select an item outside of the current window. When the window loses focus, it slides back out of view.

Specifying a second monitor

If you have a second monitor and your operating system supports it, you can choose which monitor displays a window. You can even group multiple windows together in *rafts* on other monitors.

TIP

You can create multiple instances of **Solution Explorer** and move them to another monitor. Right-click the window and choose **New Solution Explorer View**. You can return all windows back to the original monitor by double-clicking while choosing the **Ctrl** key.

Reset, name, and switch between window layouts

You can return the IDE to the original window layout for your settings collection by using the **Reset Window Layout** command. When you run this command, the following actions occur:

- All windows are moved to their default positions.
- Windows that are closed in the default window layout are closed.
- Windows that are open in the default window layout are opened.

Create and save custom layouts

Visual Studio enables you to save up to 10 custom window layouts and quickly switch between them. The following steps show how to create, save, invoke, and manage custom layouts that take advantage of multiple monitors with both docked and floating tool windows.

First, create a test solution that has two projects, each with a different optimal layout.

Create a UI project and customize the layout

1. Create a new C# **WPF App** project. Imagine that in this project, you'll be developing a user interface. You want to maximize the space for the designer window and move other tool windows out of the way.
2. If you have multiple monitors, pull the **Solution Explorer** window and the **Properties** window over to your second monitor. On a single monitor system, try closing all the windows except the designer.
3. Press **Ctrl+Alt+X** to display the **Toolbox** window. If the window is docked, drag it so that it floats somewhere where you'd like to position it.
4. Press **F5** to put Visual Studio into debugging mode. Adjust the position of the **Autos**, **Call Stack**, and **Output** debugging windows the way you want them. The layout you're about to create will apply to both editing mode and debugging mode.
5. When your layouts in both debugging mode and editing mode are how you want them, choose **Window** > **Save Window Layout**. Call this layout "Designer."

Note that your new layout is assigned the next keyboard shortcut from the reserved list of **Ctrl+Alt+1...0**.

Create a database project and layout

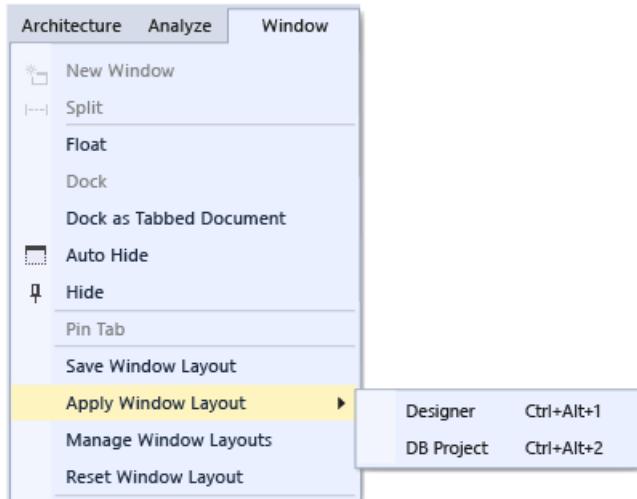
1. Add a new **SQL Server Database** project to the solution.
2. Right-click on the new project in **Solution Explorer** and choose **View in Object Explorer**. This displays the **SQL Server Object Explorer** window, which enables you to access tables, views and other objects in your database. You can either float this window or leave it docked. Adjust the other tool windows the way

you want them. For added realism, you can add an actual database, but it's not necessary for this walkthrough.

- When your layout is how you want it, from the main menu choose **Window > Save Window Layout**. Call this layout "DB Project." (We won't bother with a debug mode layout for this project.)

Switch between the layouts

To switch between layouts, use the keyboard shortcuts, or from the main menu choose **Window > Apply Window Layout**.



After applying the UI layout, note how the layout is preserved both in editing mode and in debug mode.

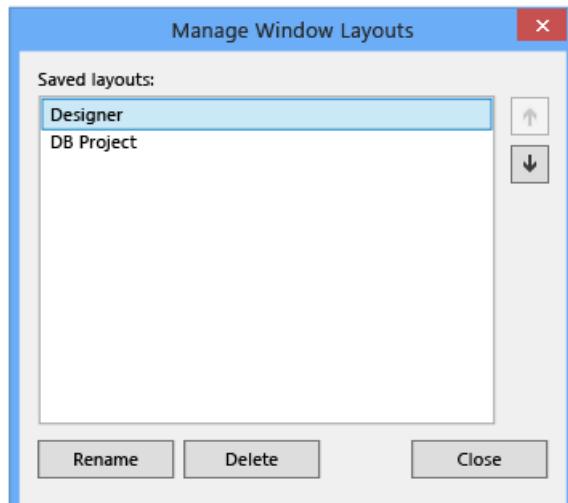
If you have a multi monitor setup at work and a single monitor laptop at home, you can create layouts that are optimized for each machine.

NOTE

If you apply a multi-monitor layout on a single-monitor system, the floating windows that you placed on the second monitor will now be hidden behind the Visual Studio window. You can bring these windows to the front by pressing **Alt + Tab**. If you later open Visual Studio with multiple monitors, you can restore the windows to their specified positions by re-applying the layout.

Manage and roam your layouts

You can remove, rename or reorder your custom layout by choosing **Window > Manage Window Layouts**. If you move a layout, the key binding is automatically adjusted to reflect the new position in the list. The bindings cannot be otherwise modified, and so you can store a maximum of 10 layouts at a time.



To remind yourself which keyboard shortcut is assigned to which layout, choose **Window > Apply Window**

Layout

These layouts automatically roam between Visual Studio editions, and also between Blend instances on separate machines, and from any Express edition to any other Express organization. However, layouts do not roam across Visual Studio, Blend and Express.

See also

- [How to: Move around in the IDE](#)

File nesting in Solution Explorer

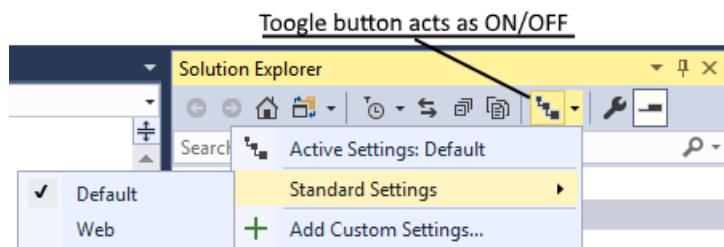
7/5/2019 • 6 minutes to read • [Edit Online](#)

Solution Explorer nests related files to help organize them and make them easier to locate. For example, if you add a Windows Forms form to a project, the code file for the form is nested below the form in **Solution Explorer**. In ASP.NET Core projects, file nesting can be taken a step further. You can choose between the file nesting presets **Off**, **Default**, and **Web**. You can also [customize how files are nested](#) or [create solution-specific and project-specific settings](#).

NOTE

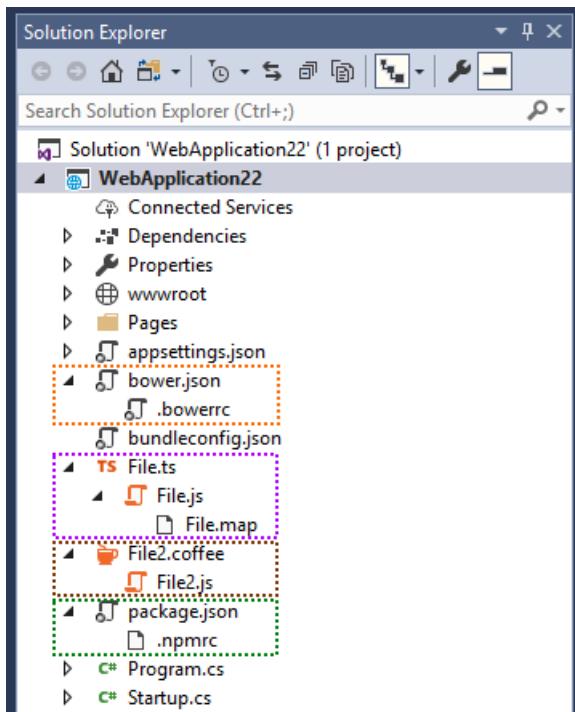
The feature is currently only supported for ASP.NET Core projects.

File nesting options



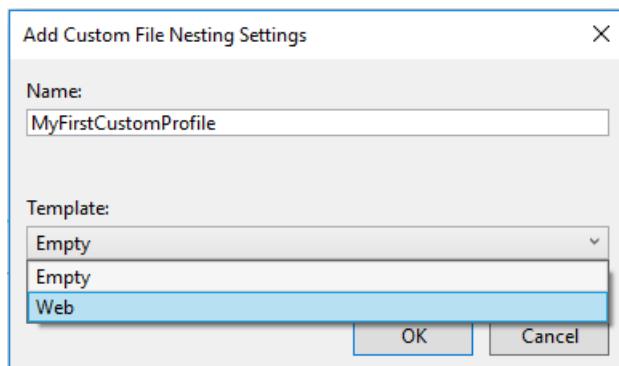
The available options for non-customized file nesting are:

- **Off**: This option gives you a flat list of files without any nesting.
- **Default**: This option gives you the default file nesting behavior in **Solution Explorer**. If no settings exist for a given project type, then no files in the project are nested. If settings exist, for example, for a web project, nesting is applied.
- **Web**: This option applies the **Web** file nesting behavior to all the projects in the current solution. It has numerous rules, and we encourage you to check it out and tell us what you think. The following screenshot highlights just a few examples of the file nesting behavior that you get with this option:



Customize file nesting

If you don't like what you get out-of-the-box, you can create your own, custom file nesting settings that instruct **Solution Explorer** how to nest files. You can add as many custom file nesting settings as you like, and you can switch between them as desired. To create a new custom setting, you can start with an empty file, or you can use the **Web** settings as your starting point:



We recommend you use **Web** settings as your starting point because it's easier to work with something that already functions. If you use the **Web** settings as your starting point, the *.filenesting.json* file looks similar to the following file:

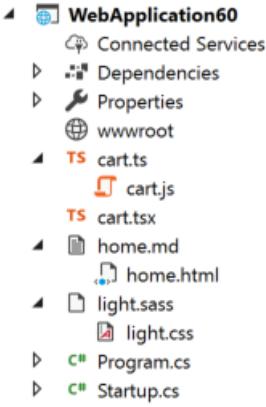
```
MyFirstCustomProfile.filenesting.json ✘ X
Schema: <No Schema Selected>
1 {{"help": "https://go.microsoft.com/fwlink/?LinkId=866610",
2 "root": true,
3
4   "dependentFileProviders": {
5     "add": {
6       "addedExtension": {},
7       "pathSegment": "...",
8       "extensionToExtension": "...",
9       "fileToFile": "...",
10      "fileSuffixToExtension": "...",
11      "allExtensions": "..."
12    }
13  }
14}
```

Let's focus on the node **dependentFileProviders** and its child nodes. Each child node is a type of rule that Visual Studio can use to nest files. For example, **having the same filename, but a different extension** is one type of rule. The available rules are:

- **extensionToExtension**: Use this type of rule to nest *file.js* under *file.ts*
- **fileSuffixToExtension**: Use this type of rule to nest *file-vsdoc.js* under *file.js*
- **addedExtension**: Use this type of rule to nest *file.html.css* under *file.html*
- **pathSegment**: Use this type of rule to nest *jquery.min.js* under *jquery.js*
- **allExtensions**: Use this type of rule to nest *file.** under *file.js*
- **fileToFile**: Use this type of rule to nest *bower.json* under *.bowerrc*

The extensionToExtension provider

This provider lets you define file nesting rules using specific file extensions. Consider the following example:



```
1  {
2    "help": "https://go.microsoft.com/fwlink/?LinkId=866610",
3    "root": true,
4
5    "dependentFileProviders": {
6      "add": {
7        "extensionToExtension": {
8          "add": {
9            ".js": [
10              ".ts",
11              ".tsx"
12            ],
13            ".css": [
14              ".scss",
15              ".sass"
16            ],
17            ".html": [
18              ".md",
19              ".markdown"
20            ],
21            ".svgz": [
22              ".svg"
23            ]
24          }
25        }
26      }
27    }
28 }
```

- *cart.js* is nested under *cart.ts* because of the first **extensionToExtension** rule
- *cart.js* is not nested under *cart.tsx* because **.ts** comes before **.tsx** in the rules, and there can only be one parent
- *light.css* is nested under *light.sass* because of the second **extensionToExtension** rule
- *home.html* is nested under *home.md* because of the third **extensionToExtension** rule

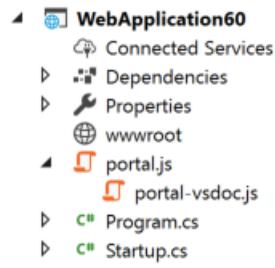
The fileSuffixToExtension provider

This provider works just like the **extensionToExtension** provider, with the only difference being that the rule looks at the suffix of the file instead of just the extension. Consider the following example:

```

1  {
2      "help": "https://go.microsoft.com/fwlink/?LinkId=866610",
3      "root": true,
4
5      "dependentFileProviders": {
6          "add": {
7              "fileSuffixToExtension": {
8                  "add": {
9                      "-vsdoc.js": [
10                          ".js"
11                      ]
12                  }
13              }
14          }
15      }
16  }

```



- *portal-vsdoc.js* is nested under *portal.js* because of the **fileSuffixToExtension** rule
- every other aspect of the rule works the same way as **extensionToExtension**

The addedExtension provider

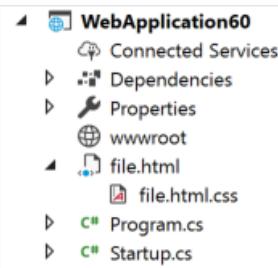
This provider nests files with an additional extension under the file without an additional extension. The additional extension can only appear at the end of the full filename.

Consider the following example:

```

1  {
2      "help": "https://go.microsoft.com/fwlink/?LinkId=866610",
3      "root": false,
4
5      "dependentFileProviders": {
6          "add": {
7              "addedExtension": {}
8          }
9      }
10 }
11

```



- *file.html.css* is nested under *file.html* because of the **addedExtension** rule

NOTE

You don't specify any file extensions for the `addedExtension` rule; it automatically applies to all file extensions. That is, any file with the same name and extension as another file plus an additional extension on the end is nested under the other file. You cannot limit the effect of this provider to just specific file extensions.

The pathSegment provider

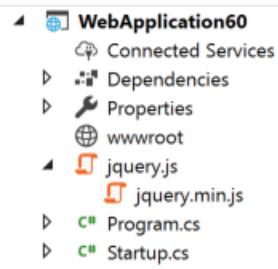
This provider nests files with an additional extension under a file without an additional extension. The additional extension can only appear at the middle of the full filename.

Consider the following example:

```

1  {
2      "help": "https://go.microsoft.com/fwlink/?LinkId=866610",
3      "root": false,
4
5      "dependentFileProviders": {
6          "add": {
7              "pathSegment": {}
8          }
9      }
10 }
11

```



- *jquery.min.js* is nested under *jquery.js* because of the **pathSegment** rule

NOTE

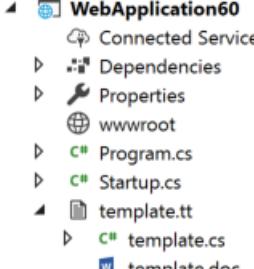
- If you don't specify any specific file extensions for the `pathSegment` rule, it applies to all file extensions. That is, any file with the same name and extension as another file plus an additional extension in the middle is nested under the other file.
- You can limit the effect of the `pathSegment` rule to specific file extensions by specifying them in the following way:

```
"pathSegment": {  
    "add": {  
        ".*": [  
            ".js",  
            ".css",  
            ".html",  
            ".htm"  
        ]  
    }  
}
```

The `allExtensions` provider

This provider lets you define file nesting rules for files with any extension but the same base file name. Consider the following example:

```
1  {  
2      "help": "https://go.microsoft.com/fwlink/?LinkId=866610",  
3      "root": false,  
4  
5      "dependentFileProviders": {  
6          "add": {  
7              "allExtensions": {  
8                  "add": {  
9                      ".*": [  
10                         ".tt"  
11                     ]  
12                 }  
13             }  
14         }  
15     }  
16 }
```

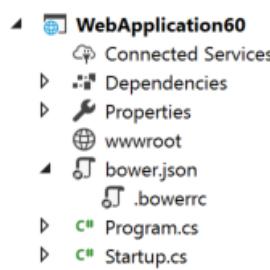


- `template.cs` and `template.doc` are nested under `template.tt` because of the **allExtensions** rule.

The `fileToFile` provider

This provider lets you define file nesting rules based on entire filenames. Consider the following example:

```
1  {  
2      "help": "https://go.microsoft.com/fwlink/?LinkId=866610",  
3      "root": false,  
4  
5      "dependentFileProviders": {  
6          "add": {  
7              "fileToFile": {  
8                  "add": {  
9                      ".bowerrc": [  
10                         "bower.json"  
11                     ]  
12                 }  
13             }  
14         }  
15     }  
16 }
```



- `.bowerrc` is nested under `bower.json` because of the **fileToFile** rule

Rule order

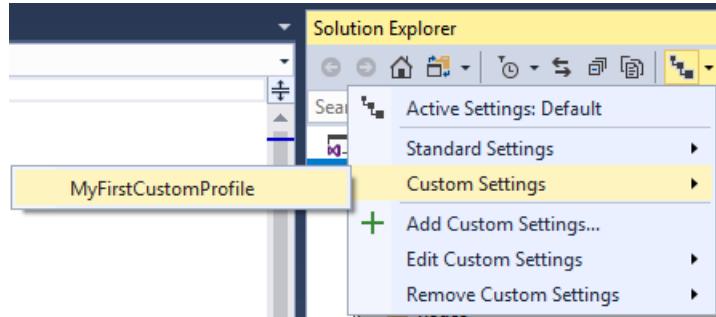
Ordering is important in every part of your custom settings file. You can change the order in which rules are

executed by moving them up or down inside of the **dependentFileProvider** node. For example, if you have one rule that makes **file.js** the parent of **file.ts** and another rule that makes **file.coffee** the parent of **file.ts**, the order in which they appear in the file dictates the nesting behavior when all three files are present. Since **file.ts** can only have one parent, whichever rule executes first wins.

Ordering is also important for rule sections themselves, not just for files within a section. As soon as a pair of files is matched with a file nesting rule, other rules further down in the file are ignored, and the next pair of files is processed.

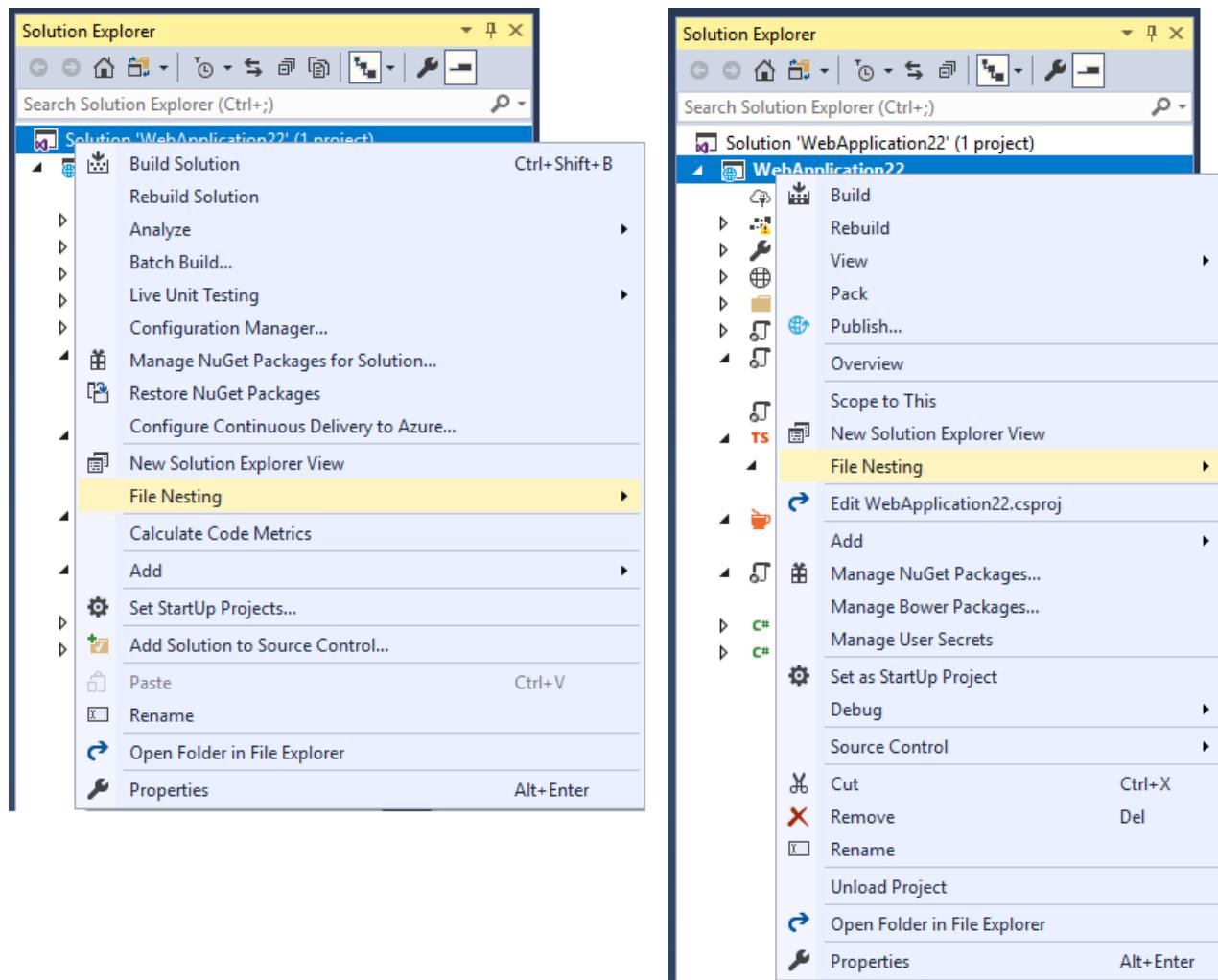
File nesting button

You can manage all settings, including your own custom settings, through the same button in **Solution Explorer**:



Create project-specific settings

You can create solution-specific and project-specific settings through the right-click menu (context menu) of each solution and project:



Solution-specific and project-specific settings are combined with the active Visual Studio settings. For example,

you may have a blank project-specific settings file, but **Solution Explorer** is still nesting files. The nesting behavior is coming from either the solution-specific settings or the Visual Studio settings. The precedence for merging file nesting settings is: Visual Studio > Solution > Project.

You can tell Visual Studio to ignore solution-specific and project-specific settings, even if the files exist on disk, by enabling the option **Ignore solution and project settings** under **Tools > Options > ASP.NET Core > File Nesting**.

You can do the opposite and tell Visual Studio to *only* use the solution-specific or the project-specific settings, by setting the **root** node to **true**. Visual Studio stops merging files at that level and doesn't combine it with files higher up the hierarchy.

Solution-specific and project-specific settings can be checked into source control, and the entire team that works on the codebase can share them.

Disable file nesting rules for a project

You can disable existing global file nesting rules for specific solutions or projects by using the **remove** action for a provider instead of **add**. For example, if you add the following settings code to a project, all **pathSegment** rules that may exist globally for this specific project are disabled:

```
"dependentFileProviders": {  
    "remove": {  
        "pathSegment": {}  
    }  
}
```

See also

- [Personalize the IDE](#)
- [Solutions and projects in Visual Studio](#)

Customize startup

10/18/2019 • 2 minutes to read • [Edit Online](#)

You can customize the startup experience for Visual Studio in several different ways, such as opening your most recent solution or just an empty development environment.

You can also show a custom start page, which is a Windows Presentation Foundation (WPF) XAML page that runs in a tool window and can run commands that are internal to Visual Studio.

To change the startup item

1. On the menu bar, choose **Tools** > **Options**.
2. Expand **Environment**, and then choose **Startup**.
3. In the **At startup** list, choose the item to be displayed after Visual Studio launches.
3. In the **On startup, open** list, choose what you want to happen after Visual Studio launches. You can choose from **Start window** (which lets you open a new or existing project), **Most recent solution**, or **Empty environment**.

To show a custom start page

You can [create your own custom start page](#) using the Visual Studio SDK, or use one that somebody else has already created. For example, you can find custom start pages at the [Visual Studio Marketplace](#).

To install a custom start page, open the .vsix file, or copy and paste the start page files into the %USERPROFILE%\Documents\Visual Studio 2017\StartPages folder on your computer.

To select which custom start page to display

1. On the menu bar, choose **Tools** > **Options**.
2. Expand **Environment**, and then choose **Startup**.
3. In the **Customize Start Page** list, choose the page that you want.

TIP

If an error in a custom start page causes Visual Studio to crash, you can open Visual Studio in safe mode and then set it to use the default start page. See [/SafeMode \(devenv.exe\)](#).

See also

- [Personalize the Visual Studio IDE](#)

Manage extensions for Visual Studio

10/18/2019 • 8 minutes to read • [Edit Online](#)

Extensions are code packages that run inside Visual Studio and provide new or improved features. Extensions may be controls, samples, templates, tools, or other components that add functionality to Visual Studio, for example, [Live Share](#) or [Visual Studio IntelliCode](#).

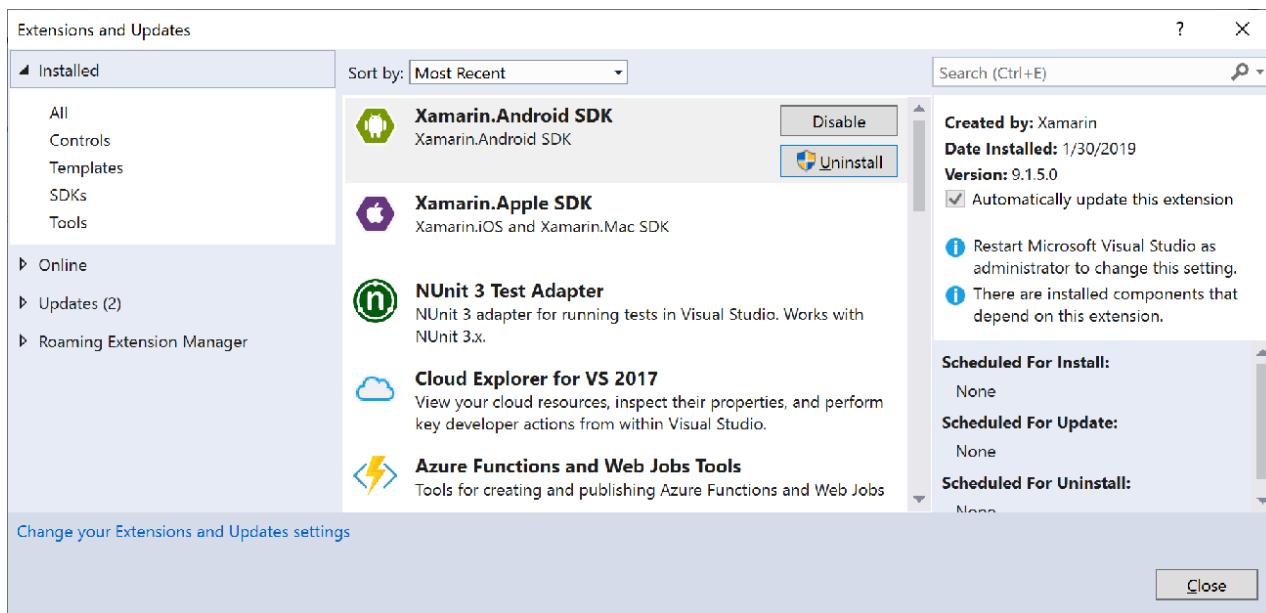
For information about creating Visual Studio extensions, see [Visual Studio SDK](#). For information about using extensions, see the individual extension page on [Visual Studio Marketplace](#).

Extensions and Updates dialog box

Use the **Extensions and Updates** dialog box to install and manage Visual Studio extensions. To open the **Extensions and Updates** dialog, choose **Tools > Extensions and Updates**, or type **Extensions** in the **Quick Launch** search box.

Manage Extensions dialog box

Use the **Manage Extensions** dialog box to install and manage Visual Studio extensions. To open the **Manage Extensions** dialog, choose **Extensions > Manage Extensions**. Or, type **Extensions** in the search box and choose **Manage Extensions**.



Find and install extensions

You can install extensions from [Visual Studio Marketplace](#) or the Extensions and Updates dialog box in Visual Studio.

To install extensions from within Visual Studio:

1. From **Tools > Extensions and Updates**, find the extension you want to install. If you know the name or

part of the name of the extension, you can search in the **Search** window.

2. Select **Download**.

The extension is scheduled for install. Your extension will be installed after all instances of Visual Studio have been closed.

If you try to install an extension that has dependencies, the installer verifies whether they're already installed. If they aren't installed, the **Extensions and Updates** dialog box lists the dependencies that must be installed before you can install the extension.

Install without using the Extensions and Updates dialog box

Extensions that have been packaged in .vsix files may be available in locations other than Visual Studio Marketplace. The **Tools > Extensions and Updates** dialog box can't detect these files, but you can install a .vsix file by double-clicking the file or selecting the file and pressing **Enter**. After that, just follow the instructions. When the extension is installed, you can use the **Extensions and Updates** dialog box to enable it, disable it, or uninstall it.

NOTE

- Visual Studio Marketplace contains both VSIX and MSI extensions. The Extensions and Updates dialog box can't enable or disable MSI-based extensions.
- If an MSI-based extension includes an *extension.vsixmanifest* file, the extension appears in the **Extensions and Updates** dialog box.

You can install extensions from [Visual Studio Marketplace](#) or the Manage Extensions dialog box in Visual Studio.

To install extensions from within Visual Studio:

1. From **Extensions > Manage Extensions**, find the extension you want to install. (If you know the name or part of the name of the extension, you can search in the **Search** window.)
2. Select **Download**.

The extension is scheduled for install. Your extension will be installed after all instances of Visual Studio have been closed.

If you try to install an extension that has dependencies, the installer verifies whether they're already installed. If they aren't installed, the **Manage Extensions** dialog box lists the dependencies that must be installed before you can install the extension.

Install without using the Manage Extensions dialog box

Extensions that have been packaged in .vsix files may be available in locations other than Visual Studio Marketplace. The **Extensions > Manage Extensions** dialog box can't detect these files, but you can install a .vsix file by double-clicking the file or selecting the file and pressing **Enter**. After that, just follow the instructions. When the extension is installed, you can use the **Manage Extensions** dialog box to enable it, disable it, or uninstall it.

NOTE

- Visual Studio Marketplace contains both VSIX and MSI extensions. The Manage Extensions dialog box can't enable or disable MSI-based extensions.
- If an MSI-based extension includes an *extension.vsixmanifest* file, the extension appears in the **Manage Extensions** dialog box.

Uninstall or disable an extension

If you want to stop using an extension, you can either disable it or uninstall it. Disabling an extension keeps it installed but unloaded. Find the extension and click **Uninstall** or **Disable**. Restart Visual Studio to unload a disabled extension.

NOTE

You can disable VSIX extensions but not extensions that were installed using an MSI. MSI-installed extensions can only be uninstalled.

Per-user and administrative extensions

Most extensions are per-user and are installed in the `%LocalAppData%\Microsoft\VisualStudio\<Visual Studio version>\Extensions\` folder. A few extensions are administrative extensions and are installed in the `<Visual Studio installation folder>\Common7\IDE\Extensions\` folder.

To protect your system against extensions that may contain errors or malicious code, you can restrict per-user extensions to load only when Visual Studio is run with normal user permissions. This means that per-user extensions are disabled when Visual Studio is run with elevated permissions.

To restrict when per-user extensions load:

1. Open the extensions options page (**Tools > Options > Environment > Extensions**).
2. Clear the **Load per user extensions when running as administrator** check box.
3. Restart Visual Studio.

Automatic extension updates

Extensions are updated automatically when a new version is available on Visual Studio Marketplace. The new version of the extension is detected and installed in the background. The next time you open Visual Studio, the new version of the extension will be running.

If you wish to disable automatic updates, you can disable the feature for all extensions or only for specific extensions.

- To disable automatic updates for all extensions, choose the **Change your Extensions and Updates settings** link in the **Tools > Extensions and Updates** dialog box. In the **Options** dialog, uncheck **Automatically update extensions**.
- To disable automatic updates for a specific extension, uncheck the **Automatically update this extension** option in the extension's details pane on the right side of the **Extensions and Updates** dialog.
- To disable automatic updates for all extensions, choose the **Change your settings for Extensions** link in the **Extensions > Manage Extensions** dialog box. In the **Options** dialog, uncheck **Automatically update extensions**.
- To disable automatic updates for a specific extension, uncheck the **Automatically update this extension** option in the extension's details pane on the right side of the **Manage Extensions** dialog.

Crash and unresponsiveness notifications

Visual Studio notifies you if it suspects that an extension was involved in a crash during a previous session. When Visual Studio crashes, it stores the exception stack. The next time Visual Studio launches, it examines the stack, starting with the leaf and working towards the base. If Visual Studio determines that a frame belongs to a module that is part of an installed and enabled extension, it shows a notification.

Visual Studio also notifies you if it suspects an extension is causing the UI to be unresponsive.

When these notifications are shown, you can ignore the notification or take one of the following actions:

- Choose **Disable this extension**. Visual Studio disables the extension and lets you know whether you need to restart your system for the disabling to take effect. You can re-enable the extension in the **Tools > Extensions and Updates** dialog box if you want.
- Choose **Disable this extension**. Visual Studio disables the extension and lets you know whether you need to restart your system for the disabling to take effect. You can re-enable the extension in the **Extensions > Manage Extensions** dialog box if you want.
- Choose **Never show this message again**.
 - If the notification concerns a crash in a previous session, Visual Studio no longer shows a notification when a crash associated with this extension occurs. Visual Studio will still show notifications when unresponsiveness can be associated with this extension, or for crashes or unresponsiveness that can be associated with other extensions.
 - If the notification concerns unresponsiveness, the integrated development environment (IDE) no longer shows a notification when this extension is associated with unresponsiveness. Visual Studio will still show crash-related notifications for this extension and crash- and unresponsiveness-related notifications for other extensions.
- Choose **Learn more** to navigate to this page.
- Choose the **X** button at the end of the notification to dismiss the notification. A new notification will appear for future instances of the extension being associated with a crash or UI unresponsiveness.

NOTE

A UI unresponsiveness or crash notification means only that one of the extension's modules was on the stack when the UI was unresponsive or when the crash occurred. It does not necessarily mean that the extension itself was the culprit. It's possible that the extension called code that's part of Visual Studio, which in turn resulted in unresponsive UI or a crash. However, the notification may still be useful if the extension which led to the UI unresponsiveness or crash is not important to you. In this case, disabling the extension avoids the UI unresponsiveness or the crash in the future, without impacting your productivity.

Samples

When you install an online sample, the solution is stored in two locations:

- A working copy is stored in the location that you specified when you created the project.
- A separate master copy is stored on your computer.

You can use the **Tools > Extensions and Updates** dialog box to perform these samples-related tasks:

You can use the **Extensions > Manage Extensions** dialog box to perform these samples-related tasks:

- List the master copies of samples that you have installed.
- Disable or uninstall the master copy of a sample.
- Install Sample Packs, which are collections of samples that relate to a technology or feature.
- Install individual online samples.
- View update notifications when source code changes are published for installed samples.

- Update the master copy of an installed sample when there is an update notification.

See also

- [Visual Studio Marketplace](#)
- [Visual Studio SDK](#)

Manage external tools

10/18/2019 • 2 minutes to read • [Edit Online](#)

You can call external tools from inside Visual Studio by using the **Tools** menu. A few default tools are available from the **Tools** menu, and you can customize the menu by adding other executables of your own.

Tools available on the Tools menu

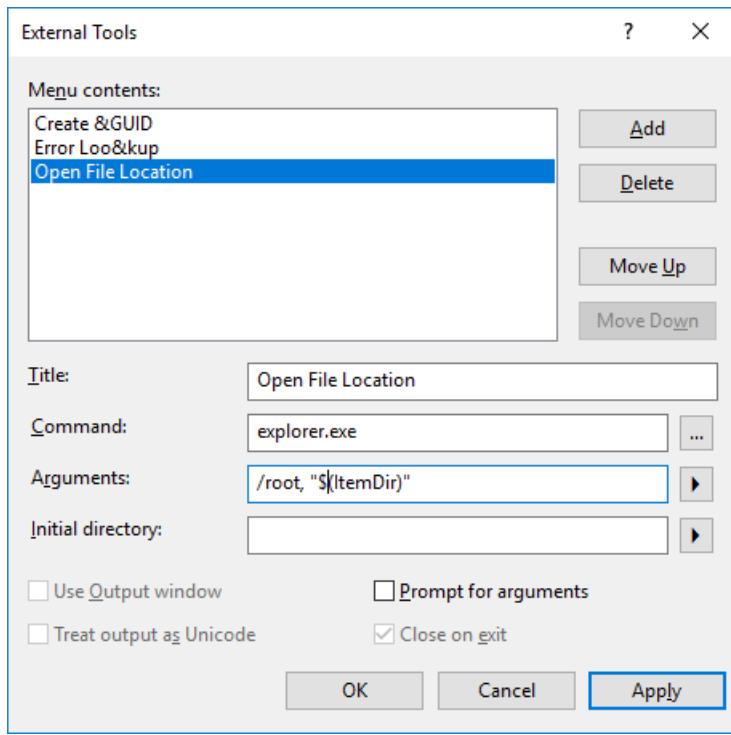
The **Tools** menu contains several built-in commands, including:

- **Extensions and Updates** to [Manage Visual Studio Extensions](#)
 - **Code Snippets Manager** to [Organize Code Snippets](#)
 - **Customize** to [Customize menus and toolbars](#)
 - **Options** to [Set a variety of different options for the Visual Studio IDE and other tools](#)
-
- **Code Snippets Manager** to [Organize Code Snippets](#)
 - **Customize** to [Customize menus and toolbars](#)
 - **Options** to [Set a variety of different options for the Visual Studio IDE and other tools](#)

Add new tools to the Tools menu

You can add an external tool to appear on the **Tools** menu.

1. Open the **External Tools** dialog box by choosing **Tools > External Tools**.
2. Click **Add**, and then fill in the information. For example, the following entry causes **Windows Explorer** to open at the directory of the file you currently have open in Visual Studio:
 - Title: `Open File Location`
 - Command: `explorer.exe`
 - Arguments: `/root, "$(ItemDir)"`



The following is a full list of arguments that can be used when defining an external tool:

NAME	ARGUMENT	DESCRIPTION
Item Path	<code>\$(ItemPath)</code>	The complete file name of the current file (drive + path + file name).
Item Directory	<code>\$(ItemDir)</code>	The directory of the current file (drive + path).
Item File Name	<code>\$(ItemFilename)</code>	The file name of the current file (file name).
Item Extension	<code>\$(ItemExt)</code>	The file name extension of the current file.
Current Line	<code>\$(CurLine)</code>	The current line position of the cursor in the code window.
Current Column	<code>\$(CurCol)</code>	The current column position of the cursor in the code window.
Current Text	<code>\$(CurText)</code>	The selected text.
Target Path	<code>\$(TargetPath)</code>	The complete file name of the item to be built (drive + path + file name).
Target Directory	<code>\$(TargetDir)</code>	The directory of the item to be built.
Target Name	<code>\$(TargetName)</code>	The file name of the item to be built.
Target Extension	<code>\$(TargetExt)</code>	The file name extension of the item to be built.

NAME	ARGUMENT	DESCRIPTION
Binary Directory	<code>\$(BinDir)</code>	The final location of the binary that is being built (defined as drive + path).
Project Directory	<code>\$(ProjectDir)</code>	The directory of the current project (drive + path).
Project File Name	<code>\$(ProjectFileName)</code>	The file name of the current project (drive + path + file name).
Solution Directory	<code>\$(SolutionDir)</code>	The directory of the current solution (drive + path).
Solution File Name	<code>\$(SolutionFileName)</code>	The file name of the current solution (drive + path + file name).

NOTE

The IDE status bar displays the **Current Line** and **Current Column** variables to indicate where the insertion point is located in the active **Code Editor**. The **Current Text** variable returns the text or code selected at that location.

See also

- [C/C++ build tools](#)

Sign in to Visual Studio

10/25/2019 • 3 minutes to read • [Edit Online](#)

You can personalize and optimize your development experience in Visual Studio if you set your personalization account by signing in to the IDE.

NOTE

This topic applies to Visual Studio on Windows. For Visual Studio for Mac, see [Sign in to Visual Studio for Mac](#).

Why should I sign in to Visual Studio?

When you sign in, you enrich your Visual Studio experience. For example, after you sign in, you can [synchronize your settings](#) across devices, extend a trial, and automatically connect to an Azure service, to name a few.

Here's a full list of what you can expect and what you can do after you sign in:

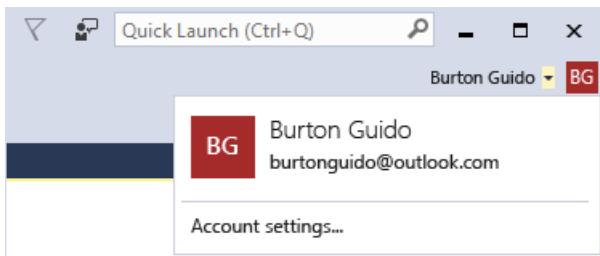
- **Access to the Visual Studio Dev Essentials program** - This program includes free software offerings, training, support, and more. See [Visual Studio Dev Essentials](#) for more information.
- **Synchronize your Visual Studio settings** - Settings that you customize, such as key bindings, window layout, and color theme, apply immediately when you sign in to Visual Studio on any device. See [Synchronized settings in Visual Studio](#).
- **Unlock the Visual Studio Community edition** - If your Community edition installation prompts you for a license, sign in to the IDE to unlock yourself.
- **Extend the Visual Studio trial period** - You can use Visual Studio Professional or Visual Studio Enterprise for an additional 90 days, instead of being limited to the trial period of 30 days.
- **Unlock Visual Studio if you use an account that's associated with a Visual Studio subscription or an Azure DevOps organization**. See [How to unlock Visual Studio](#).
- **Automatically connect to services such as Azure and Azure DevOps Services** in the IDE without prompting again for credentials for the same account.

How to sign in to Visual Studio

When you open Visual Studio for the first time, you're asked to sign in and provide some basic registration information. You should choose a Microsoft account or a work or school account that best represents you. If you don't have any of these accounts, you can create a Microsoft account for free. See [How do I sign up for a Microsoft account?](#)

Next, choose the UI settings and color theme that you want to use in Visual Studio. Visual Studio remembers these settings and synchronizes them across all Visual Studio environments you have signed in to. For a list of the settings that are synchronized, see [Synchronized settings](#). You can change the settings later if you open the **Tools** > **Options** menu in Visual Studio.

After you provide the settings, Visual Studio starts, and you're signed in and ready to get started. To verify whether you're signed in, look for your name in the upper-right corner of the Visual Studio environment.



Unless you sign out, you're automatically signed in to Visual Studio whenever you start it, and any changes to synchronized settings are automatically applied. To sign out, choose the down arrow next to your profile name in the upper-right corner of the Visual Studio environment, choose the **Account settings** command, and then choose the **Sign out** link. To sign in again, choose the **Sign in** command in the upper-right corner of the Visual Studio environment.

To change your profile information

1. Go to **File > Account Settings** and choose the **Manage Visual Studio profile** link.
2. In the browser window, choose **Edit profile** and change the settings that you want.
3. When you're done, choose **Save changes**.

Troubleshooting

If you encounter any problems while signing in, please see the [Accounts support](#) page to get help.

See also

- [How to unlock Visual Studio](#)
- [Visual Studio IDE overview](#)
- [Sign in \(Visual Studio for Mac\)](#)
- [Activation \(Visual Studio for Mac\)](#)

Work with multiple user accounts

10/18/2019 • 5 minutes to read • [Edit Online](#)

If you have multiple Microsoft accounts and/or work or school accounts, you can add them all to Visual Studio so that you can access the resources from any account without having to sign in to it separately. Azure, Application Insights, Azure DevOps, and Office 365 services all support the streamlined sign-in experience.

After you add multiple accounts on one machine, that set of accounts roams with you if you sign in to Visual Studio on another machine.

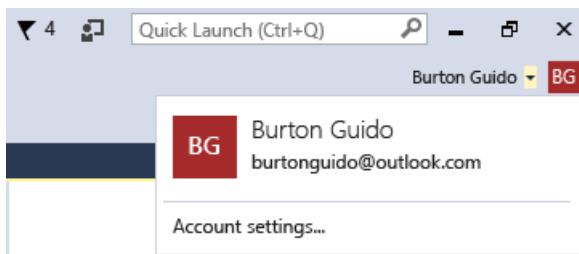
NOTE

Although the account names roam, the credentials do not. You'll be prompted to enter credentials for those other accounts the first time you attempt to use their resources on a new machine.

This article shows you how to add multiple accounts to Visual Studio. It also shows you how to see the resources accessible from those accounts in places such as the **Add Connected Service** dialog, **Server Explorer**, and **Team Explorer**.

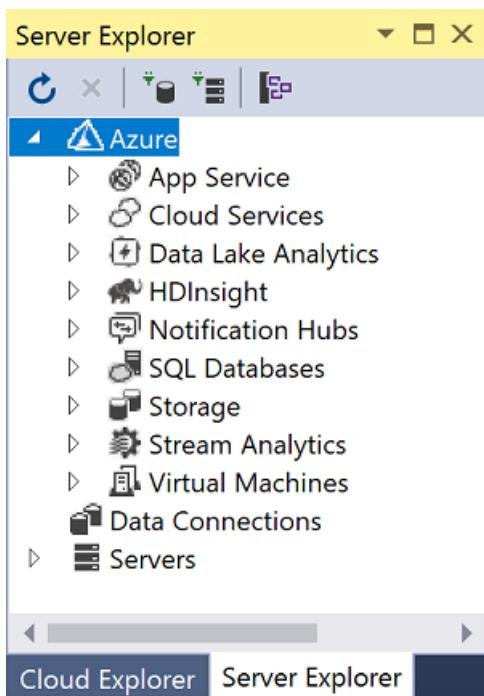
Sign in to Visual Studio

Sign into Visual Studio with a Microsoft account or an organizational account. You should see your user name appear in the upper corner of the window, similar to this:

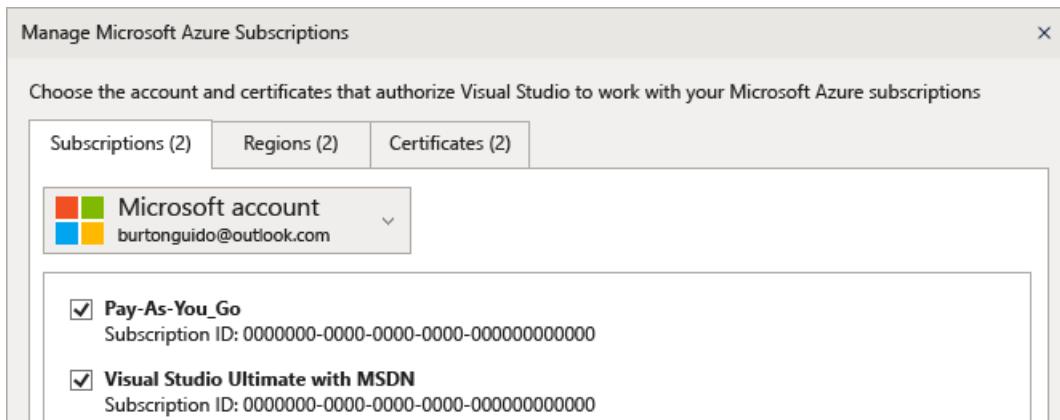


Access your Azure account in Server Explorer

To open Server Explorer, choose **View > Server Explorer** (or, if you're using the "General" environment settings, press **Ctrl+Alt+S**). Expand the **Azure** node and notice that it contains the resources available in the Azure account that's associated with the account that you used to sign in to Visual Studio. It looks similar to the following image:



The first time you use Visual Studio on any specific device, the dialog only shows the subscriptions registered under the account that you signed in with. You can access resources for any of your other accounts directly from **Server Explorer** by right-clicking on the **Azure** node, choosing **Manage and Filter Subscriptions**, and then adding your accounts from the account picker control. You can then choose another account, if desired, by clicking the down arrow and choosing from the list of accounts. After choosing the account, you can customize which subscriptions under that account to display in **Server Explorer**.



The next time you open **Server Explorer**, the resources for that subscription are displayed.

Access your Azure account via Add Connected Service dialog

1. Open an existing project, or create a new project.
2. Choose the project node in **Solution Explorer**, and then right-click and choose **Add > Connected Service**.

The **Add Connected Service** wizard appears and shows you the list of services in the Azure account that's associated with your Visual Studio personalization account. You don't have to sign in separately to Azure. However, you do need to sign in to the other accounts the first time you attempt to access their resources from a different machine.

Access Azure Active Directory in a Web project

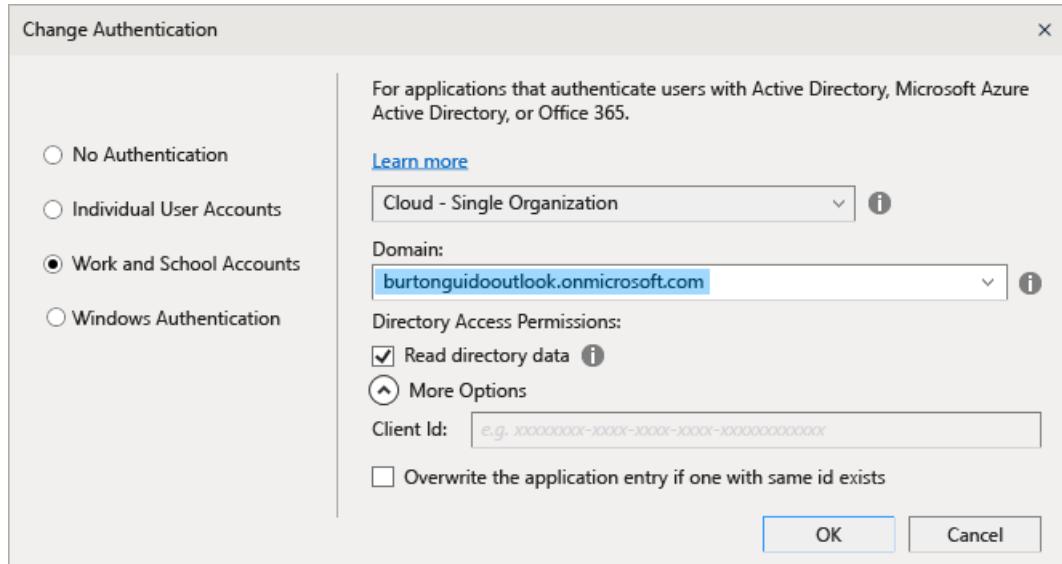
Azure Active Directory (AAD) enables support for end-user single sign-in in ASP.NET MVC web apps or AD authentication in web API services. Domain authentication is different from individual user account authentication. Users that have access to your Active Directory domain can use their existing AAD accounts to connect to your

web applications. Office 365 apps can also use domain authentication.

To see this in action, create a new **ASP.NET Core Web Application** project. In the **New ASP.NET Core Web Application** dialog box, choose the **Web Application** template, and then choose **Change Authentication**.

To see this in action, create a new **ASP.NET Core Web Application** project. On the **Create a new ASP.NET Core Web Application** page, choose the **Web Application** template, and then choose **Change** under **Authentication**.

The **Change Authentication** dialog box appears where you can choose what kind of authentication to use in your application.



For more information about the different kinds of authentication in ASP.NET, see [Create ASP.NET web projects in Visual Studio](#).

Access your Azure DevOps organization

From the main menu, choose **Team > Manage Connections** to open the **Team Explorer - Connect** window. Choose **Manage Connections > Connect to a Project**. In the **Connect to a Project** dialog, select a project from the list (or select **Add TFS Server** and enter the URL to your server). When you select a URL, you're logged in without having to reenter your credentials.

For more information, see [Connect to projects in Team Explorer](#).

Add an additional account to Visual Studio

To add an additional account to Visual Studio:

1. Choose **File > Account Settings**.
2. Under **All Accounts**, choose **Add an account**.
3. On the **Sign in to your account** page, select the account or choose **Use another account**. Follow the prompts to enter the new account credentials.

(Optional) Now you can go to **Server Explorer** and see the Azure services associated with the account you just added. In **Server Explorer**, right-click on the **Azure** node and choose **Manage and Filter Subscriptions**. Choose the new account by clicking the drop-down arrow next to the current account, and then choose which subscriptions you want to display in **Server Explorer**. You should see all the services associated with the specified subscription. Even though you're not currently signed into Visual Studio with the second account, you are signed in to that account's services and resources. The same is true for **Project > Add Connected Service** and **Team > Connect to Team Foundation Server**.

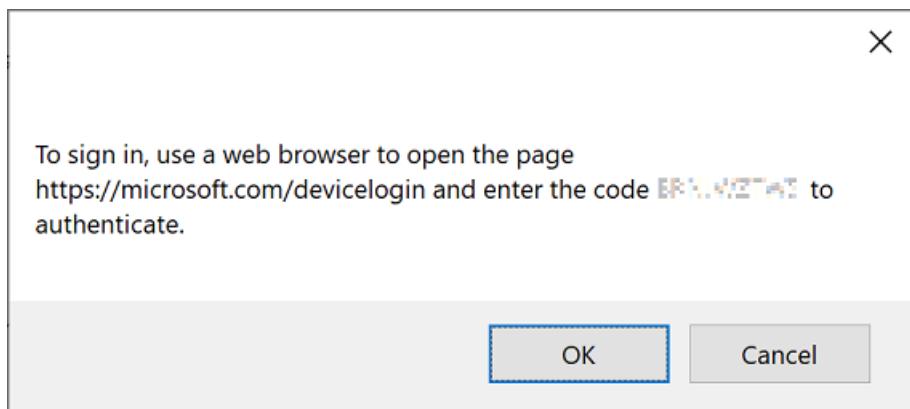
Add an account using device code flow

In some cases, you can't sign in or add an account in the regular manner. This can happen if Internet Explorer is blocked for some reason, or if your network is behind a firewall. To work around this, you can enable *device code flow* to add an account or reauthenticate your account. Device code flow lets you sign in using a different browser or on a different machine—either physical or virtual (VM).

To sign in using device code flow:

1. Open the **Accounts** page under **Tools > Options > Environment**, and then select **Enable device code flow when adding or re-authenticating an account**. Choose **OK** to close the options pages.
2. Choose **File > Account Settings** to open the account management page.
3. Choose **Add an account** under **All Accounts**.

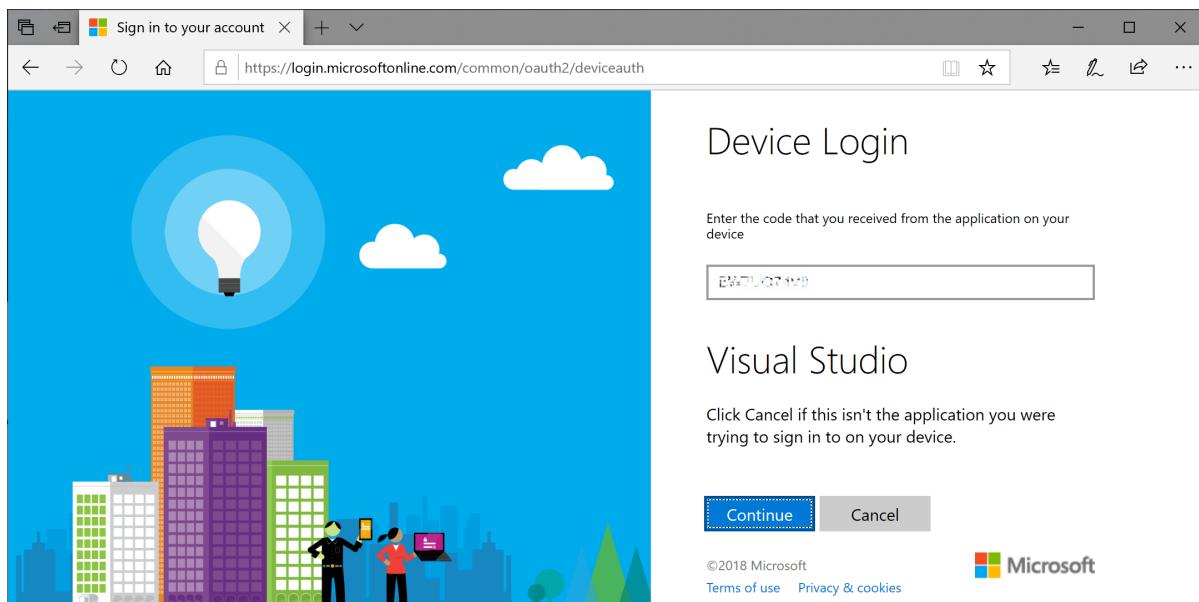
A dialog box shows you a URL and a code to paste into a web browser.



4. Press **Ctrl+C** to copy the text of the dialog, and then choose **OK** to close the dialog. Paste the text you copied into a text editor such as Notepad. This makes it easier to copy the code in the next step.
5. Navigate to the device login URL on the machine or web browser you want to use to sign in to Visual Studio, and then paste or enter the code you copied into the box that says **Code**.

The **Visual Studio** app name should appear further down on the page.

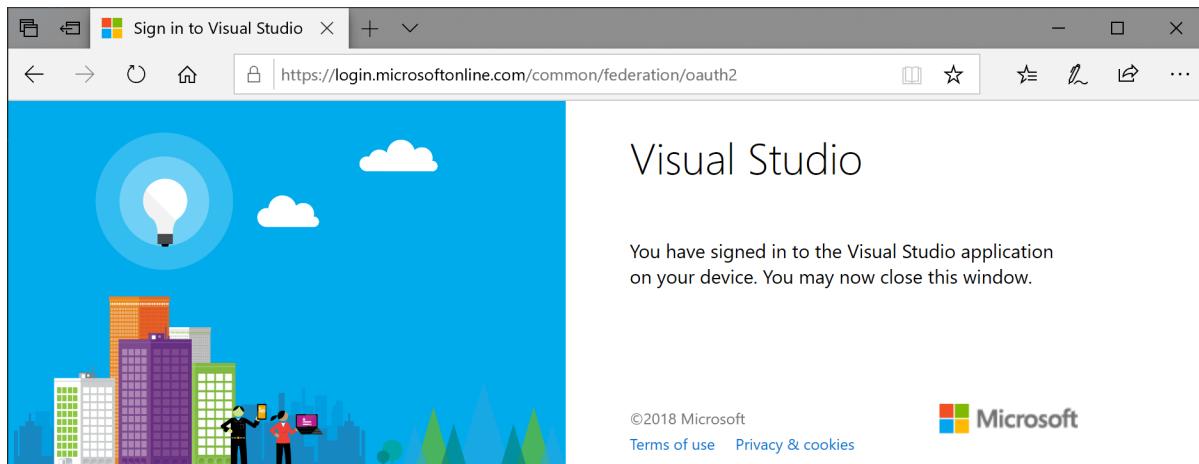
6. Under **Visual Studio**, choose **Continue**.



7. Follow the prompts to enter your account credentials.

A page appears telling you that you've signed into Visual Studio on your device, and that you can close the

browser window.



8. Go back to the account management page in Visual Studio and you'll see the newly added account listed under **All Accounts**. Choose **Close**.

See also

- [Sign in to Visual Studio](#)
- [Sign in to Visual Studio for Mac](#)

How to: Unlock Visual Studio

8/9/2019 • 2 minutes to read • [Edit Online](#)

You can evaluate Visual Studio for free up to 30 days. Signing into the IDE extends the trial period to 90 days. To continue using Visual Studio, unlock the IDE by either:

- using an online subscription
- entering a product key

To unlock Visual Studio using an online subscription

To unlock Visual Studio using a Visual Studio subscription or an Azure DevOps organization associated with a Microsoft account, or a work or school account:

1. Choose the **Sign in** button in the upper-right corner of the IDE (or go to **File > Account Settings** to open the **Account Settings** dialog and choose the **Sign in** button).
2. Enter the credentials for either a Microsoft account or a work or school account. Visual Studio finds a Visual Studio subscription or an Azure DevOps organization associated with your account.

IMPORTANT

Visual Studio automatically looks for associated online subscriptions when you connect to an Azure DevOps organization from the **Team Explorer** tool window. When you connect to an Azure DevOps organization, you can sign in using both Microsoft and work or school accounts. If an online subscription exists for that user account, Visual Studio will automatically unlock the IDE for you.

To unlock Visual Studio with a product key

1. Select **File > Account Settings** to open the **Account Settings** dialog, and then choose the **License with a Product Key** link.
2. Enter the product key in the space provided.

TIP

Pre-release versions of Visual Studio do not have product keys. You must sign in to the IDE to use pre-release versions.

Address license problem states

Update stale licenses

You might have seen the following message that says that your license is going stale in Visual Studio. It reads, "Your license has gone stale and must be updated."



Your license has gone stale and must be updated.
Check for an update license to continue using this product.

[Check for an updated license](#)

This message indicates that while your subscription may still be valid, the license token Visual Studio uses to keep

your subscription up to date hasn't been refreshed and has gone stale due to one of the following reasons:

- You have not used Visual Studio or have had no internet connection for an extended period of time.
- You signed out of Visual Studio.

Before the license token goes stale, Visual Studio first shows a warning message that asks you to reenter your credentials.

If you do not reenter your credentials, the token starts to go stale and the **Account Settings** dialog tells you how many days you have left before your token will fully expire. After your token expires, you must reenter your credentials for the account before you can continue using Visual Studio.

IMPORTANT

If you are using Visual Studio for extended periods in environments with limited or no internet access, you should use a product key to unlock Visual Studio to avoid interruption.

Update expired licenses

If your subscription has expired completely and you no longer have access rights to Visual Studio, you must renew your subscription or add another account that has a subscription. To see more information about the license you are using, go to **File > Account Settings** and look at the license information on the right side of the dialog. If you have another subscription associated with a different account, add that account to the **All Accounts** list on the left side of the dialog box by selecting the **Add an account** link.

See also

- [Sign in to Visual Studio](#)

Optimize Visual Studio performance

10/18/2019 • 2 minutes to read • [Edit Online](#)

This article provides some suggestions to try if you find that Visual Studio is running slowly. You can also take a look at [Visual Studio performance tips and tricks](#) for more suggestions on how to improve performance.

Upgrade Visual Studio

If you are currently using Visual Studio 2015, download [Visual Studio 2017](#) or [Visual Studio 2019](#) for free to check out its improved performance. Solutions load two to three times faster than in Visual Studio 2015, with performance improvements in other areas too. Visual Studio 2017 and Visual Studio 2019 are side-by-side compatible with Visual Studio 2015, so you won't lose anything by trying it out.

If you're already using Visual Studio 2017, make sure you are running version 15.6 or later. Data shows that solutions load up to two or three times faster in version 15.6. Download it [here](#).

Extensions and tool windows

You may have extensions installed that are slowing Visual Studio down. For help on managing extensions to improve performance, see [Change extension settings to improve performance](#).

Similarly, you may have tool windows that are slowing Visual Studio down. For help on managing tool windows, see [Change tool window settings to improve performance](#).

Hardware

If you're thinking about upgrading your hardware, a solid state drive (SSD) has more effect on performance than additional RAM or a faster CPU.

If you do add an SSD, for optimal performance install Windows on that drive as opposed to a hard disk drive (HDD). The drive location of your Visual Studio solutions does not seem to matter as much.

Additionally, do not run your solution from a USB drive. Copy it to your HDD or SSD.

Help us improve

Your feedback helps us improve. Use the **Report a Problem** feature to "record" a trace and send to us. Select the feedback icon next to **QuickLaunch**, or select **Help > Send Feedback > Report a Problem** from the menu bar. For more information, see [How to report a problem with Visual Studio](#).

See also

- [Performance tips and tricks](#)
- [Visual Studio blog - Load solutions faster with Visual Studio 2017 version 15.6](#)

Optimize Visual Studio startup time

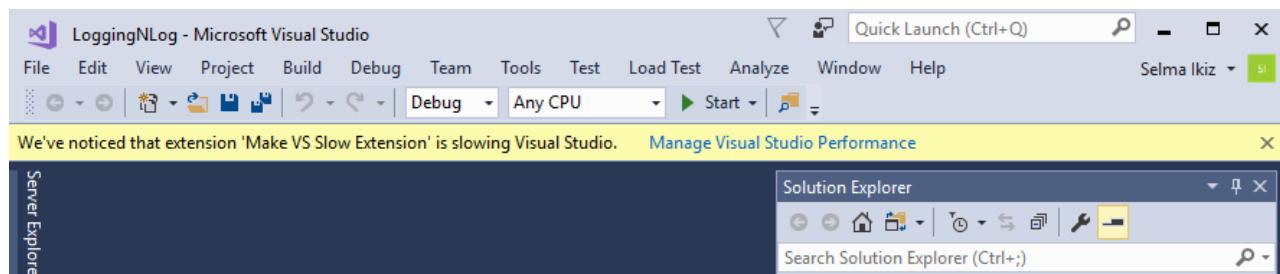
10/18/2019 • 2 minutes to read • [Edit Online](#)

Visual Studio is designed to start up as quickly and efficiently as possible. However, certain Visual Studio extensions and tool windows can adversely affect startup time when they are loaded. You can control the behavior of slow extensions and tool windows in the **Manage Visual Studio Performance** dialog box. For more general tips on improving performance, see [Visual Studio performance tips and tricks](#).

Startup behavior

To avoid extending startup time, Visual Studio loads extensions using an *on demand* approach. This behavior means that extensions don't open immediately after Visual Studio starts, but on an as-needed basis. Also, because tool windows left open in a prior Visual Studio session can slow startup time, Visual Studio opens tool windows in a more intelligent way to avoid impacting startup time.

If Visual Studio detects slow startup, a pop-up message appears, alerting you to the extension or tool window that's causing the slowdown. The message provides a link to the **Manage Visual Studio Performance** dialog box. You can also access this dialog box by choosing **Help > Manage Visual Studio Performance** from the menu bar.

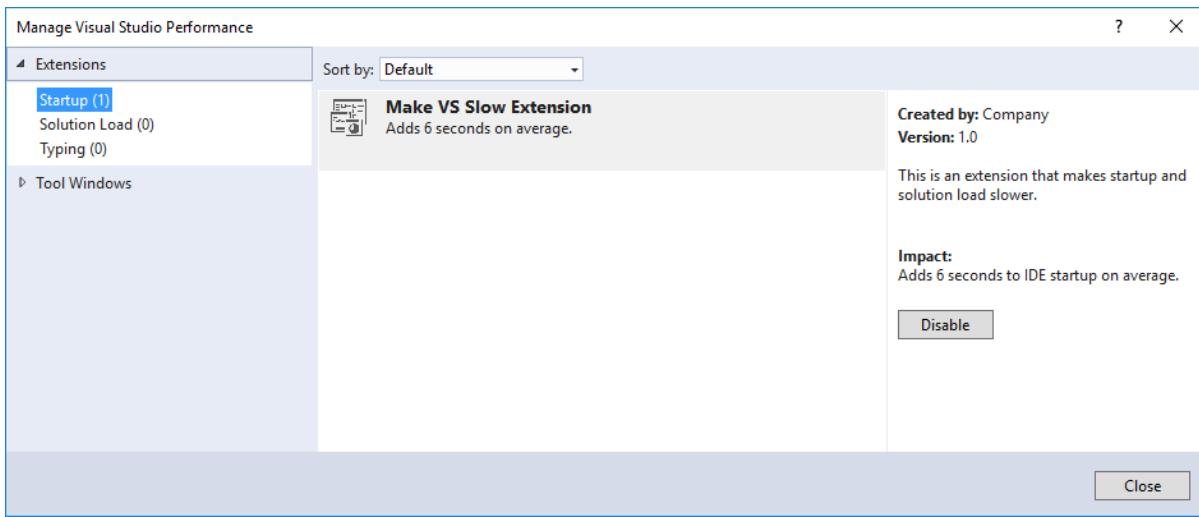


The dialog box lists the extensions and tools windows that are affecting startup performance. You can change extension and tool window settings to improve startup performance.

To change extension settings to improve startup, solution load, and typing performance

1. Open the **Manage Visual Studio Performance** dialog box by choosing **Help > Manage Visual Studio Performance** from the menu bar.

If an extension is slowing down Visual Studio startup, solution loading, or typing, the extension appears in the **Manage Visual Studio Performance** dialog box under **Extensions > Startup** (or **Solution Load** or **Typing**).



- Choose the extension you want to disable, then choose the **Disable** button.

You can always re-enable the extension for future sessions by using the **Extension Manager** or the **Manage Visual Studio Performance** dialog box.

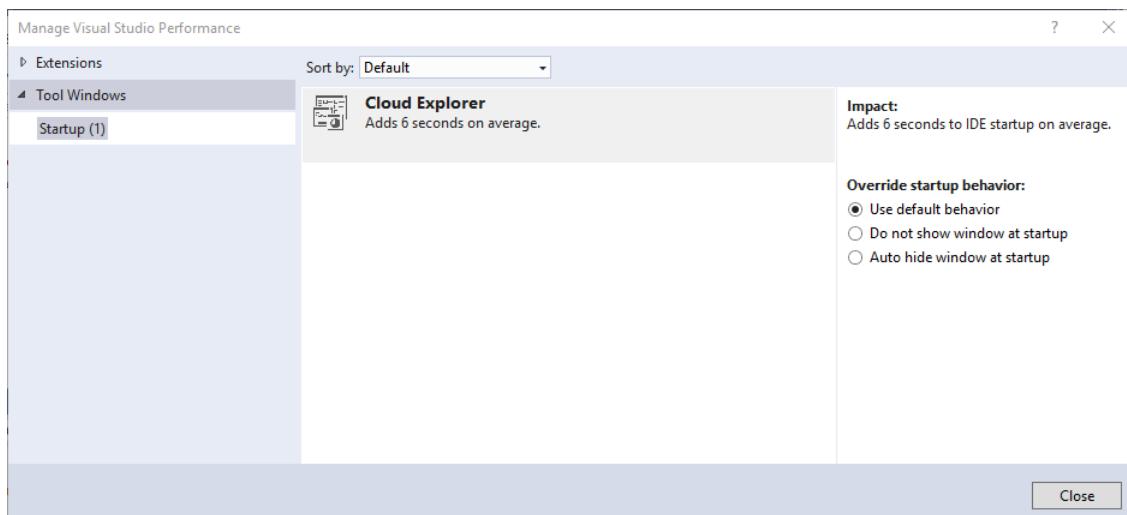
To change tool window settings to improve startup time

- Open the **Manage Visual Studio Performance** dialog box by choosing **Help > Manage Visual Studio Performance** from the menu bar.

If a tool window is slowing down Visual Studio startup, the tool window appears in the **Manage Visual Studio Performance** dialog box under **Tool Windows > Startup**.

- Choose the tool window you want to change the behavior for.
- Choose one of the following three options:

- Use default behavior:** The default behavior for the tool window. Keeping this option selected will not improve startup performance.
- Do not show window at startup:** The specified tool window is always closed when you open Visual Studio, even if you left it open in a previous session. You can open the tool window from the appropriate menu when you need it.
- Auto hide window at startup:** If a tool window was left open in a previous session, this option collapses the tool window's group at startup to avoid initializing the tool window. This option is a good choice if you use a tool window often. The tool window is still available, but no longer negatively affects Visual Studio startup time.



NOTE

Some earlier versions of Visual Studio 2017 had a feature called **lightweight solution load**. In current versions, large solutions that contain managed code load much faster than previously, even without lightweight solution load.

See also

- [Optimize Visual Studio performance](#)
- [Visual Studio performance tips and tricks](#)
- [Visual Studio blog - Load solutions faster with Visual Studio 2017 version 15.6](#)

Filtered solutions in Visual Studio

10/18/2019 • 3 minutes to read • [Edit Online](#)

Large development teams often collaborate by using a single large solution with many projects. However, individual developers typically work on a small subset of these projects. To improve performance when opening large solutions, Visual Studio 2019 introduced *solution filtering*. Solution filtering lets you open a solution with only selective projects loaded. Loading a subset of projects in a solution decreases solution load, build, and test run time, and enables more focused review.

The following features are available:

- You can get to code faster by opening a solution without loading any of its projects. After the solution opens, you can selectively choose which projects to load.
- When you reopen a solution, Visual Studio remembers which projects were loaded in your previous session and only loads those projects.
- You can create a solution filter file to save one or more project-load configurations or share the configuration with teammates.

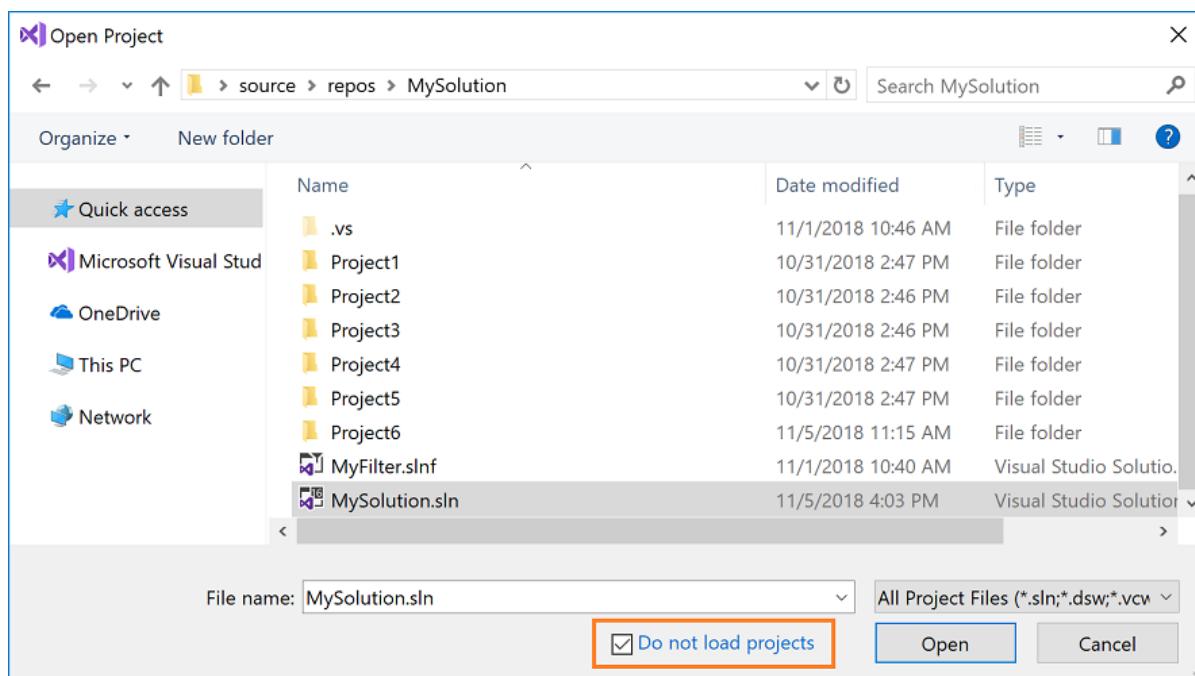
Open a filtered solution

You can open a solution without loading any of its projects directly from the **Open Project** dialog or through the [command line](#).

Open Project dialog

To open a solution without loading any of its projects by using the **Open Project** dialog:

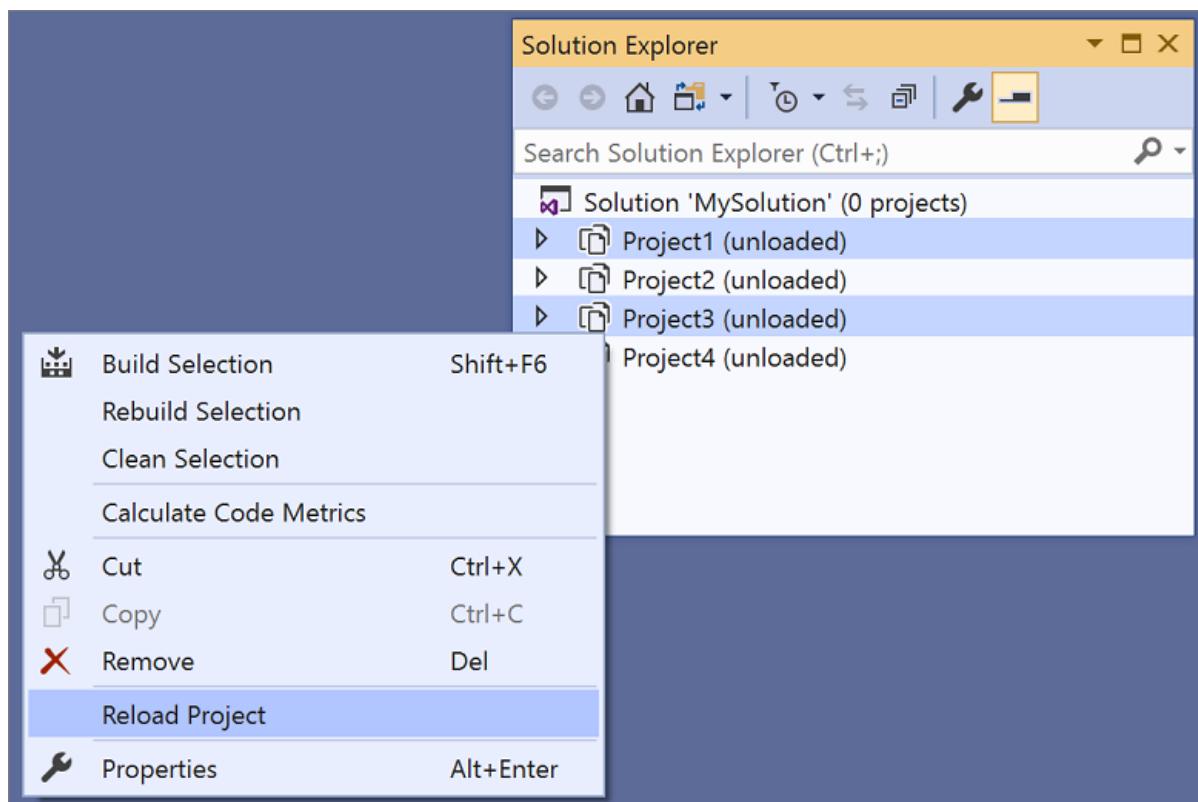
1. Choose **File > Open > Project/Solution** from the menu bar.
2. In the **Open Project** dialog, select the solution, and then select **Do not load projects**.



3. Choose **Open**.

The solution opens with all of its projects unloaded.

4. In **Solution Explorer**, select the projects you want to load (press **Ctrl** while clicking to select more than one project), and then right-click on the project and choose **Reload Project**.



Visual Studio will remember which projects are loaded the next time you open the solution locally.

Command line

(New in Visual Studio 2019 version 16.1.)

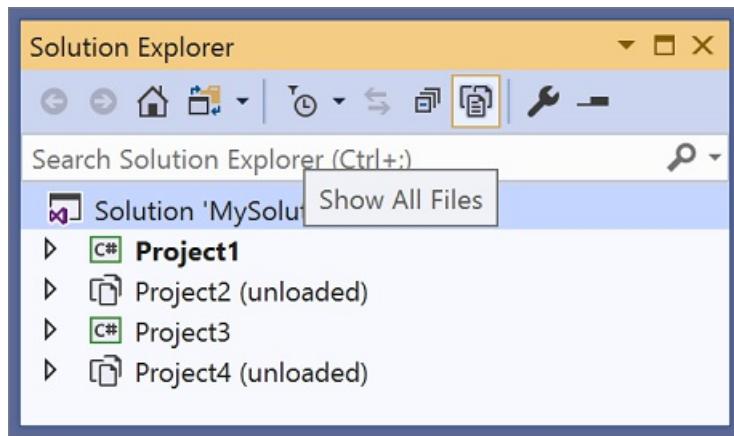
To open a solution without loading any of its projects from the command line, use the `/donotloadprojects` switch as shown in the following example:

```
devenv /donotloadprojects MySln.sln
```

Toggle unloaded project visibility

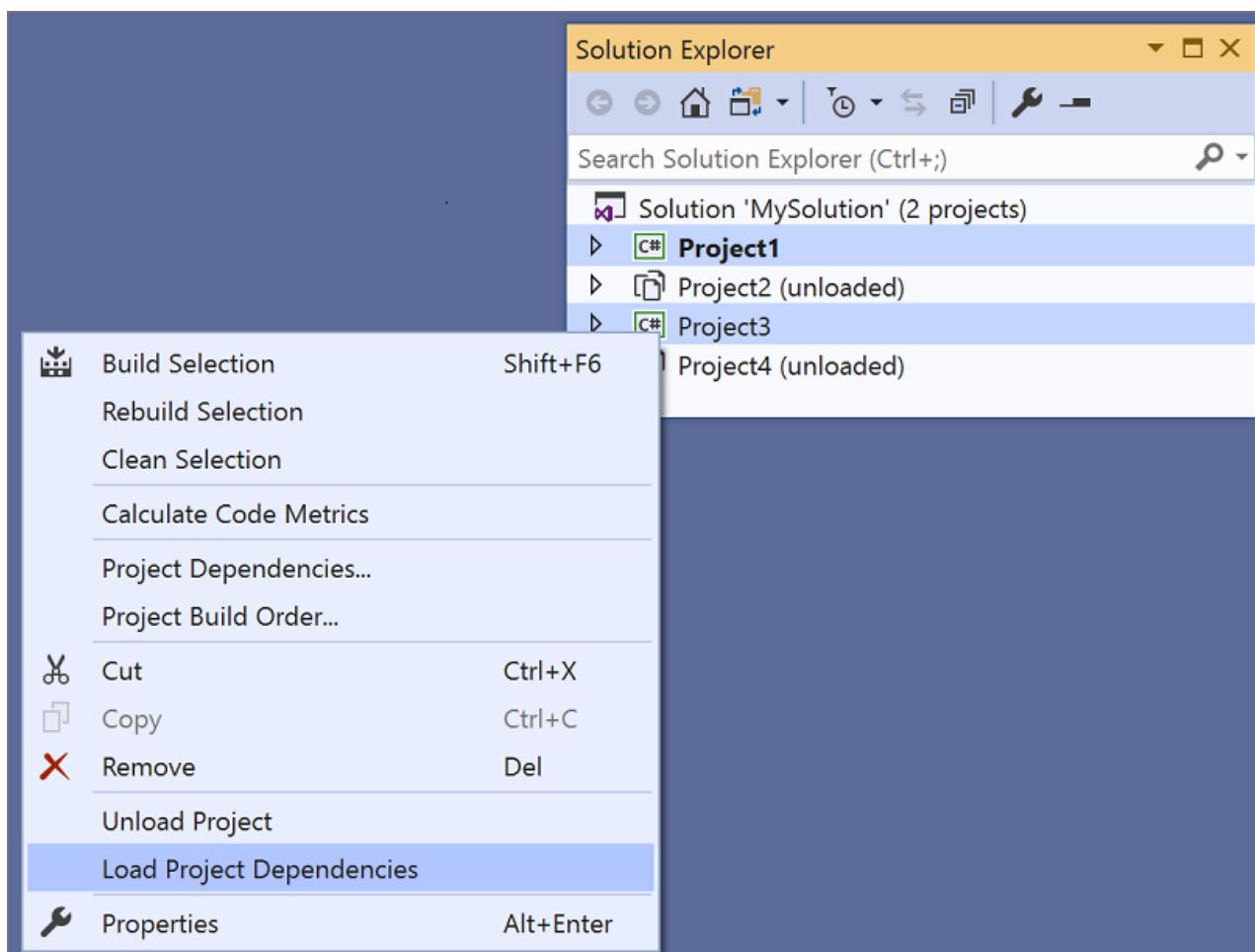
You can choose to see either all the projects in the solution or just the loaded ones using one of the following choices in **Solution Explorer**:

- Right-click on your solution and select **Show Unloaded Projects** or **Hide Unloaded Projects**.
- Select the solution node to enable the **Show All Files** button; then, click the button to toggle the visibility of unloaded projects.



Load project dependencies

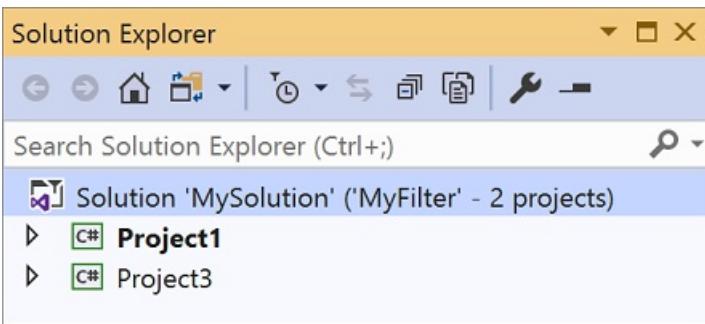
In a solution where only selected projects are loaded, you may not have all of a project's project dependencies loaded. Use the **Load project dependencies** menu option to ensure that any projects that a project depends on are also loaded. Right-click on one or more loaded projects in **Solution Explorer** and choose **Load project dependencies**.



Solution filter files

If you want to share your project-load configuration or commit it to source control, you can create a solution filter file (it has the extension `.slnf`). When you open a solution filter file, the solution opens in Visual Studio with the specified projects loaded and all the unloaded projects hidden. You can [toggle](#) to view the unloaded projects.

Solution filter files are visually differentiated from regular solution files by the additional funnel glyph in the icon next to the solution in **Solution Explorer**. The name of the filter and the number of loaded projects are also shown next to the solution name.

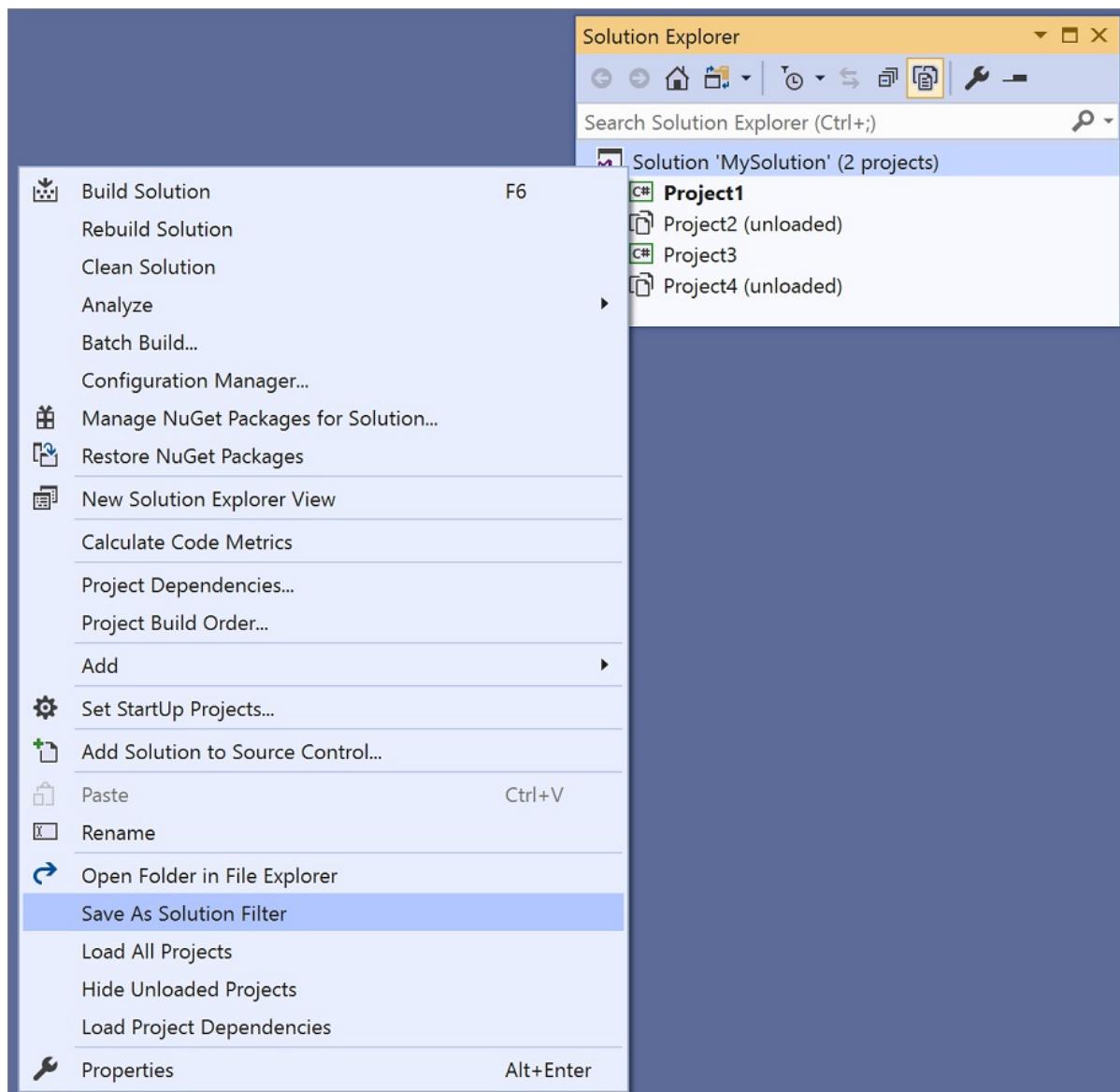


NOTE

If new projects are added to the original solution after you create the solution filter file, they appear as unloaded projects in **Solution Explorer**.

Create a solution filter file

1. In **Solution Explorer**, right-click on the solution and select **Save As Solution Filter**.



2. Choose a name and location for the solution filter file.

After you create a solution filter file, it's added to your **Recent Projects and Solutions** list for easy access:

What would you like to do?

Open recent



MyFilter.slnf

C:\Users\source/repos\MySolution



MySolution.sln

C:\Users\source/repos\MySolution

See also

- [Customize file nesting in Solution Explorer](#)
- [Optimize Visual Studio performance](#)

Visual Studio performance tips and tricks

10/18/2019 • 6 minutes to read • [Edit Online](#)

Visual Studio performance recommendations are intended for low memory situations, which may occur in rare cases. In these situations, you can optimize certain Visual Studio features that you may not be using. The following tips are not intended as general recommendations.

NOTE

If you're having difficulty using the product because of memory issues, let us know through the [feedback tool](#).

Use a 64-bit OS

If you upgrade your system from a 32-bit version of Windows to a 64-bit version, you expand the amount of virtual memory available to Visual Studio from 2 GB to 4 GB. This enables Visual Studio to handle significantly larger workloads, even though it is 32-bit process.

For more information, see [Memory limits](#) and [Use /LARGEADDRESSAWARE on 64-bit Windows](#).

Disable automatic file restore

Visual Studio automatically reopens documents that were left open in the previous session. This can prolong the times it takes to load a solution by up to 30% or more, depending on the project type and the documents being opened. Designers like Windows Forms and XAML, and some JavaScript and typescript files, can be slow to open.

Visual Studio notifies you in a yellow bar when automatic document restore is causing a solution to load significantly slower. You can disable automatic file reopening by following these steps:

1. Select **Tools > Options** to open the **Options** dialog box.
2. On the **Projects and Solution > General** page, deselect **Reopen documents on solution load**.

If you disable automatic file restore, a quick way to navigate to files you want to open is by using one of the [Go To](#) commands:

- For the general **Go To** functionality, select **Edit > Go To > Go To All**, or press **Ctrl+T**.
- Jump to the last edit location in a solution using **Edit > Go To > Go To Last Edit Location**, or by pressing **Ctrl+Shift+Backspace**.
- Use **Go To Recent File** to see a list of recently visited files in a solution. Select **Edit > Go To > Go To Recent File**, or press **Ctrl+1, Ctrl+R**.

Configure debugging options

If you are typically running low on memory during debugging sessions, you can optimize performance by making one or more configuration changes.

- **Enable Just My Code**

The simplest optimization is to enable the **Just My Code** feature, which only loads symbols for your project. Enabling this feature can result in a significant memory saving for debugging managed applications (.NET). This option is already enabled by default in some project types.

To enable **Just My Code**, choose **Tools > Options > Debugging > General**, and then select **Enable Just My Code**.

- **Specify symbols to load**

For native debugging, loading symbol files (.pdb) is expensive in terms of memory resources. You can configure your debugger symbol settings to conserve memory. Typically, you configure the solution to only load modules from your project.

To specify symbol loading, choose **Tools > Options > Debugging > Symbols**.

Set the options to **Only specified modules** instead of **All modules** and then specify which modules you care to load. While debugging, you can also right-click specific modules in the **Modules** window to explicitly include a module in the symbol load. (To open the window while debugging, choose **Debug > Windows > Modules**.)

For more information, see [Understand symbol files](#).

- **Disable Diagnostic Tools**

It is recommended that you disable CPU profiling after use. This feature can consume large amounts of resources. Once CPU profiling is enabled, this state is persisted across subsequent debug sessions, so it's worth explicitly turning it off when done. You may save some resources by disabling the diagnostic tools while debugging if you do not need the provided features.

To disable the **Diagnostic Tools**, start a debugging session, choose **Tools > Options > Enable Diagnostic Tools**, and deselect the option.

For more information, see [Profiling Tools](#).

Disable tools and extensions

Some tools or extensions can be turned off to improve performance.

TIP

You can often isolate performance issues by turning off extensions one at a time and rechecking performance.

Managed language service (Roslyn)

For information about .NET Compiler Platform ("Roslyn") performance considerations, see [Performance considerations for large solutions](#).

- **Disable full solution analysis**

Visual Studio performs analysis on your entire solution in order to provide a rich experience about errors before invoking a build. This feature is useful to identify errors as soon as possible. However, for large solutions, this feature can consume significant memory resources. If you're experiencing memory pressure or similar issues, you can disable this experience to free up these resources. By default, this option is enabled for Visual Basic and disabled for C#.

To disable **Full Solution Analysis**, choose **Tools > Options > Text Editor**, then select either **Visual Basic** or **C#**. Choose **Advanced** and deselect **Enable full solution analysis**.

- **Disable CodeLens**

Visual Studio performs a **Find All References** task on each method as it is displayed. CodeLens provides features such as the inline display of the number of references. The work is performed in a separate process such as *ServiceHub.RoslynCodeAnalysisService32*. In large solutions, or on resource-constrained systems,

this feature can have a significant impact on performance. If you're experiencing memory issues, for example, when loading a large solution on a 4-GB machine, or high CPU usage for this process, you can disable CodeLens to free up resources.

To disable **CodeLens**, choose **Tools > Options > Text Editor > All Languages > CodeLens**, and deselect the feature.

NOTE

CodeLens is available in the Professional and Enterprise editions of Visual Studio.

Other tools and extensions

• Disable Extensions

Extensions are additional software components added to Visual Studio that provide new functionality or extend existing functionality. Extensions can often be a source of memory resource issues. If you're experiencing memory resource problems, try disabling extensions one at a time to see how it impacts the scenario or workflow.

To disable extensions, go to **Tools > Extensions and Updates**, and disable a particular extension.

To disable extensions, go to **Extensions > Manage Extensions**, and disable a particular extension.

• Disable XAML Designer

The XAML designer is enabled by default, but only consumes resources if you open a *.xaml* file. If you work with XAML files but do not wish to use the designer functionality, disable this feature to free up some memory.

To disable **XAML Designer**, go to **Tools > Options > XAML Designer > Enable XAML Designer**, and deselect the option.

• Remove workloads

You can use the Visual Studio Installer to remove workloads that are no longer used. This action can streamline the startup and runtime cost by skipping packages and assemblies that aren't needed anymore.

Force a garbage collection

The CLR uses a garbage collection memory management system. In this system, sometimes memory is used by objects that are no longer needed. This state is temporary; the garbage collector will release this memory based on its performance and resource usage heuristics. You can force the CLR to collect any unused memory by using a hotkey in Visual Studio. If there is a significant amount of garbage waiting for collection and you force a garbage collection, you should see the memory usage of the *devenv.exe* process drop in **Task Manager**. It's rarely necessary to use this method. However, after an expensive operation has completed (such as a full build, debug session, or a solution open event), it can help you determine how much memory is really being used by the process. Because Visual Studio is mixed (managed & native), it's occasionally possible for the native allocator and the garbage collector to compete for limited memory resources. Under conditions of high memory usage, it may help to force the garbage collector to run.

To force a garbage collection, use the hotkey: **Ctrl+Alt+Shift+F12, Ctrl+Alt+Shift+F12** (press it twice).

If forcing garbage collection reliably makes your scenario work, file a report through the Visual Studio feedback tool as this behavior is likely to be a bug.

For a detailed description of the CLR garbage collector, see [Fundamentals of garbage collection](#).

See also

- [Optimize Visual Studio performance](#)
- [Load solutions faster \(Visual Studio blog\)](#)

Accessibility features of Visual Studio

10/25/2019 • 2 minutes to read • [Edit Online](#)

In addition to accessibility features and utilities in Windows, the following features make Visual Studio more accessible for people with disabilities:

- Toolbar button and text enlargement
- Text size options in the editors
- Color customization in the editors
- Keyboard shortcut customization
- Auto-completion for methods and parameters

NOTE

This topic applies to Visual Studio on Windows. For Visual Studio for Mac, see [Accessibility for Visual Studio for Mac](#).

For more information, see the following topics:

- [How to: Set IDE accessibility options](#)
- [How to: Use the keyboard exclusively](#)
- [Default keyboard shortcuts](#)
- [Accessibility tips and tricks](#)
- [How to: Change fonts and colors](#)

IMPORTANT

The information on this page might apply only to users who license Microsoft products in the United States. If you obtained this product outside of the United States, visit the [Microsoft Accessibility](#) website for a list of Microsoft support services telephone numbers and addresses. You can contact your subsidiary to find out whether the type of products and services described on this page are available in your area. Information about accessibility is also available in other languages.

TIP

To learn more about recent accessibility updates, see the [Accessibility improvements in Visual Studio 2017 version 15.3 blog post](#).

See also

- [Accessibility products and services from Microsoft](#)

How to: Set IDE accessibility options

8/28/2019 • 4 minutes to read • [Edit Online](#)

Visual Studio includes features that make it easier for people who have low vision to read and for people who have limited dexterity to write. For example, you can change the size and color of text in editors, change the size of text and buttons on toolbars, and change settings to help you complete a function or statement.

In addition, Visual Studio supports Dvorak keyboard layouts, which make the most frequently typed characters more accessible. You can also customize the default keyboard shortcuts available with Visual Studio. For more information, see [Identify and customize keyboard shortcuts](#).

NOTE

The dialog boxes and menu commands you see might differ from those described here, which can vary depending on your active settings or edition. To change your settings, choose **Import and Export Settings** on the **Tools** menu. For more information, see [Reset settings](#).

TIP

To learn more about recent accessibility updates, see the [Accessibility improvements in Visual Studio 2017 version 15.3](#) blog post.

Editors, dialogs, and tool windows

By default, dialog boxes and tool windows in Visual Studio use the same font size and colors as the operating system. The color settings for the frame of the IDE, dialog boxes, toolbars, and tool windows are based a color scheme: light or dark. You can change the current color theme in the [Options dialog box: Environment > General](#).

You can also display pop-up windows in the Code view of the editor. These windows can prompt you with available members on the current object and the parameters to complete a function or statement. These windows can be helpful if you have difficulty typing. However, they interfere with focus in the code editor, which can be problematic for some users.

Here's how to turn off the pop-up windows:

1. From the **Tools** menu, choose **Options**.
2. Choose **Text Editor > All Languages > General**.
3. Clear the **Auto list members** and **Parameter information** checkboxes.

You can rearrange the windows in the integrated development environment (IDE) to best suit the way you work. You can dock, float, hide, or automatically hide each tool window. For more information about how to change window layouts, see [Customize window layouts](#).

Change the size of text

You can change the settings for text-based tool windows, such as the **Command** window, **Immediate** window, and **Output** window by using **Tools > Options > Environment > Fonts and Colors**.

When you select **[All Text Tool Windows]** in the **Show settings for** drop-down list, the default setting is listed as **Default** in the **Item foreground** and **Item background** drop-down lists. Choose the **Custom** button to

change these settings.

You can also change the settings for how text is displayed in the editor. Here's how.

1. From the **Tools** menu, choose **Options**.
2. Choose **Environment > Fonts and Colors**.
3. Select an option on the **Show settings for** drop-down menu.

To change the font size for text in an editor, choose **Text Editor**.

To change the font size for text in text-based tool windows, choose **[All Text Tool Windows]**.

To change the font size for ToolTip text in an editor, choose **Editor Tooltip**.

To change the font size for text in statement completion pop-ups, choose **Statement Completion**.

4. From **Display items**, select **Plain Text**.
5. In **Font**, select a new font type.
6. In **Size**, select a new font size.

TIP

To reset the text size for text-based tool windows and editors, choose **Use Defaults**.

7. Choose **OK**.

Change the colors that are used in the IDE

You can choose to change the default colors for text, margin indicators, white space, and code elements in the editor. Here's how.

1. From the **Tools** menu, choose **Options**.
2. In the **Environment** folder, choose **Fonts and Colors**.
3. In **Show settings for**, choose **Text Editor**.
4. From **Display items**, select an item whose display you need to change, such as **Plain Text**, **Indicator Margin**, **Visible White Space**, **HTML Attribute Name**, or **XML Attribute**.
5. Select display settings from the following options: **Item foreground**, **Item background**, and **Bold**.
6. Choose **OK**.

TIP

To use high contrast colors for all application windows on your operating system, press **Left Alt+Left Shift+PrtScn**. If Visual Studio is open, close and then reopen it to fully implement high contrast colors.

Toolbars

To improve toolbar usability and accessibility, you can add text to toolbar buttons.

To assign text to toolbar buttons

1. From the **Tools** menu, choose **Customize**.
2. In the **Customize** dialog box, select the **Commands** tab.

3. Select **Toolbar**, and then choose the toolbar name that contains the button you intend to display text for.
4. In the list, select the command you intend to change.
5. Choose **Modify Selection**.
6. Choose **Image and Text**.

To modify the displayed text in a button

1. Re-select **Modify Selection**.
2. Adjacent to In **Name**, insert provide a new caption for the selected button.

See also

- [Accessibility features of Visual Studio](#)
- [Accessibility for Visual Studio for Mac](#)
- [Resources for designing accessible applications](#)

Accessibility tips and tricks for Visual Studio

8/9/2019 • 4 minutes to read • [Edit Online](#)

Visual Studio has built-in accessibility features that are compatible with screen readers and other assistive technologies. Whether you want to use keyboard shortcuts to navigate the IDE, or use high-contrast themes to improve visibility, you'll find several tips & tricks on this page about how to do so.

We also cover how to use annotations to reveal useful information about your code, and how to set sound cues for build and breakpoint events.

NOTE

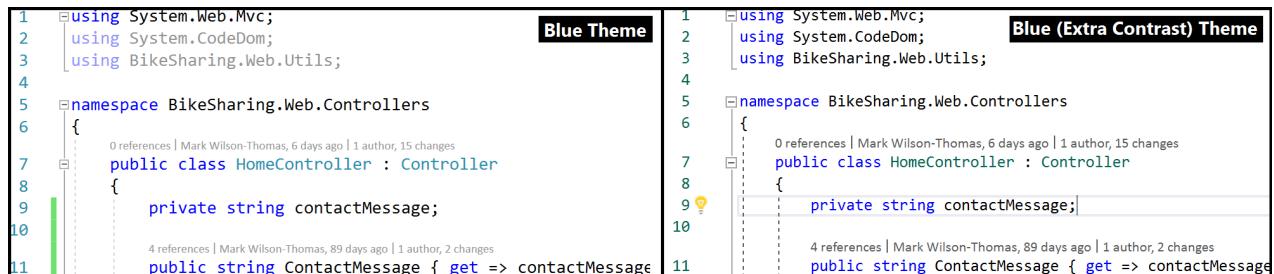
This topic applies to Visual Studio on Windows. For Visual Studio for Mac, see [Accessibility for Visual Studio for Mac](#).

Save your IDE settings

You can customize your IDE experience by saving your window layout, keyboard mapping scheme, and other preferences. For more information, see [Personalize the Visual Studio IDE](#).

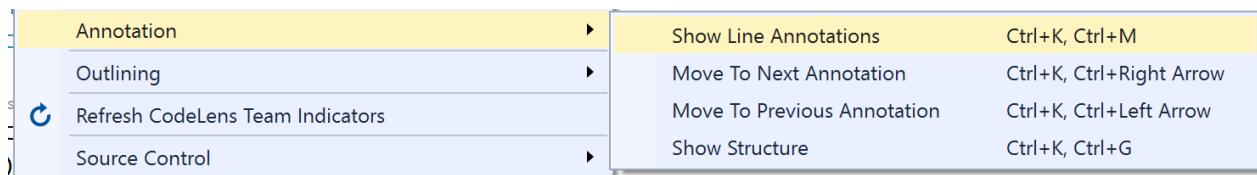
Modify your IDE for high-contrast viewing

For some folks, some colors are more difficult to see. If you want more contrast as you code but don't want to use the typical "High Contrast" themes, we now offer a "Blue (Extra Contrast)" theme.



Use annotations to reveal useful information about your code

The Visual Studio editor includes many text "adornments" that let you know about characteristics and features at particular points on a line of code, such as screwdriver and light bulb icons, error and warning "squiggles", bookmarks, and so on. You can use the "Show Line Annotations" command set to help you discover and then navigate between these adornments.



Access toolbars by using keyboard shortcuts

The Visual Studio IDE has toolbars as do many tool windows. The following keyboard shortcuts help you access them.

FEATURE	DESCRIPTION	KEYBOARD SHORTCUT
IDE toolbars	Select the first button on the Standard toolbar.	Alt, Ctrl+Tab
Tool window toolbars	Move focus to the toolbars in a tool window. NOTE: This works for most tool windows, but only when the focus is in a tool window. Also, you must choose the SHIFT key before the ALT key. In some tool windows, such as Team Explorer, you must hold the SHIFT key for a moment before choosing the ALT key.	Shift+Alt
Toolbars	Go to the first item in the next toolbar (when a toolbar has focus).	Ctrl+Tab

Other useful keyboard shortcuts

Some other useful keyboard shortcuts include the following.

FEATURE	DESCRIPTION	KEYBOARD SHORTCUT
IDE	Switch High Contrast on and off. NOTE: Standard Windows keyboard shortcut	Left Alt+Left Shift+PrtScn
Dialog box	Select or clear the check box option in a dialog box. NOTE: Standard Windows keyboard shortcut	Spacebar
Context menus	Open a context (right-click) menu. NOTE: Standard Windows keyboard shortcut	Shift+F10
Menus	Quickly access a menu item by using its accelerator keys. Choose the Alt key followed by the underlined letters in a menu to activate the command. For example, to view the Open Project dialog box in Visual Studio, you would choose Alt+F+O+P . NOTE: Standard Windows keyboard shortcut	Alt + [letter]
Search box	Use the search feature in Visual Studio.	Ctrl+Q

FEATURE	DESCRIPTION	KEYBOARD SHORTCUT
Toolbox window	Move among Toolbox tabs.	Ctrl+Up arrow and Ctrl+Down arrow
Toolbox window	Add a control from the Toolbox to a form or designer.	Enter
Options dialog box: Environment > Keyboard	Delete a key combination entered in the Press shortcut keys option.	Backspace
Notifications tool window	Open the Notifications tool window by using two keyboard shortcut key combinations, one followed by the other. Then, view a notification by using the arrow keys to select it.	Ctrl+\ Ctrl+N

NOTE

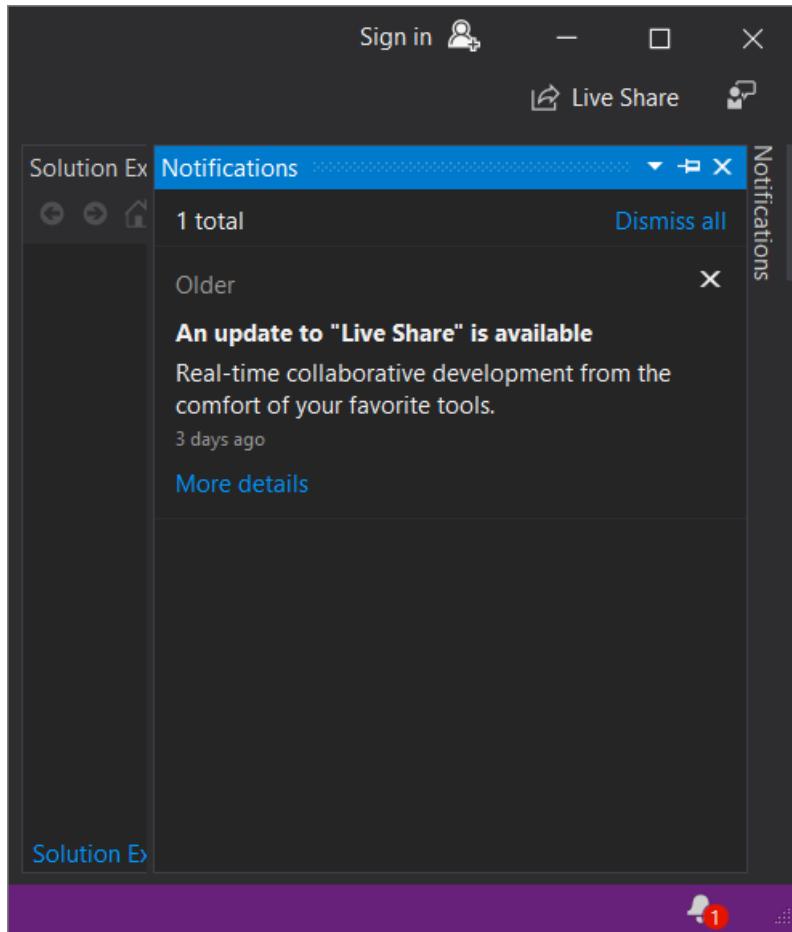
The dialog boxes and menu commands you see might differ from those described in Help, depending on your active settings or edition.

Access notifications by using keyboard shortcuts

When a notification appears in the IDE, here's how you can access the Notifications window by using keyboard shortcuts:

- From anywhere in the IDE, press the following two keyboard shortcuts in sequence, one after the other:
Ctrl+ and then **Ctrl+N**.

The **Notifications** window opens.



2. Use either the **Tab** key or the arrow keys to select a notification.

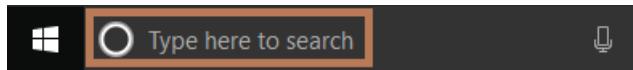
Use the Sound applet to set build and breakpoint cues

You can use the Sound applet in Windows to assign a sound to Visual Studio program events. Specifically, you can assign sounds to the following program events:

- Breakpoint hit
- Build canceled
- Build failed
- Build succeeded

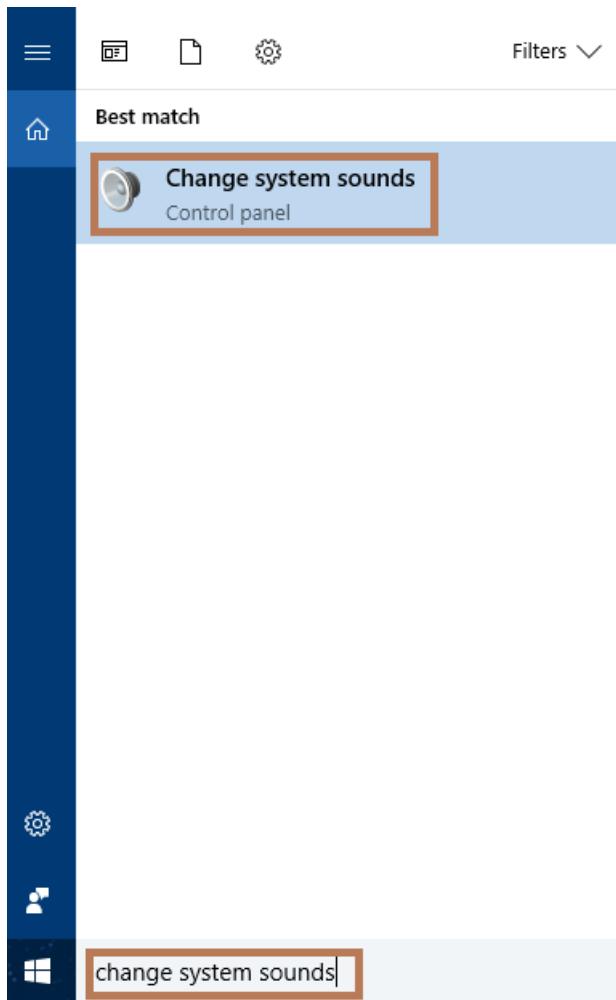
Here's how:

1. In the **Search** box on a computer running Windows 10, type **Change system sounds**.

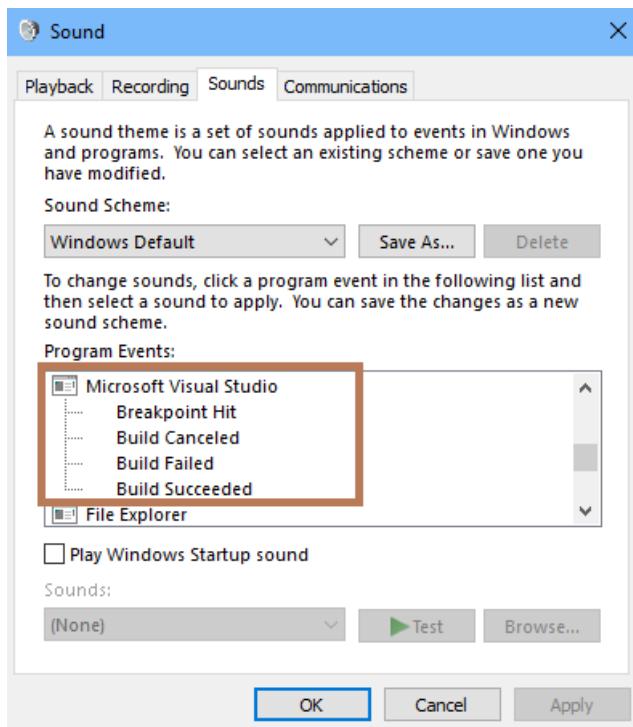


(Alternatively, if you have Cortana enabled, say "Hey Cortana", and then say "Change system sounds".)

2. Double-click **Change system sounds**.



3. In the **Sound** dialog box, click the **Sounds** tab.
4. In **Program Events**, scroll to **Microsoft Visual Studio**, and then select the sounds that you want to apply to the events that you choose.



5. Click **OK**.

TIP

To learn more about accessibility updates, see the [Accessibility improvements in Visual Studio 2017 version 15.3](#) blog post.

See also

- [Accessibility features of Visual Studio](#)
- [How to: Customize menus and toolbars in Visual Studio](#)
- [Personalize the Visual Studio IDE](#)
- [Accessibility \(Visual Studio for Mac\)](#)
- [Microsoft Accessibility](#)

Accessibility products and services from Microsoft

10/25/2019 • 2 minutes to read • [Edit Online](#)

Microsoft is committed to making its products and services easier for everyone to use. This page provides information about the features, products, and services that make Microsoft Windows more accessible for people with disabilities.

IMPORTANT

The information on this page might apply only to users who license Microsoft products in the United States. If you obtained this product outside of the United States, visit the [Microsoft Accessibility](#) website for a list of Microsoft support services telephone numbers and addresses. You can contact your subsidiary to find out whether the type of products and services described on this page are available in your area. Information about accessibility is available in other languages.

Accessibility features of Windows

The Windows operating system has many built-in accessibility features that are useful for individuals who have difficulty typing or using a mouse, are blind or have low vision, or who are deaf or hard-of-hearing. The features are installed during Setup. For more information about these features, see Help in Windows and the [Microsoft Accessibility](#) website.

Free step-by-step tutorials

Microsoft offers a series of step-by-step tutorials that provide detailed procedures for adjusting the accessibility options and settings on your computer. This information is presented in a side-by-side format so that you can learn how to use the mouse, the keyboard, or a combination of both.

To find step-by-step tutorials for Microsoft products, see the [Microsoft Accessibility](#) website.

Assistive technology products for Windows

A wide variety of assistive technology products are available to make computers easier to use for people with disabilities. You can search a catalog of assistive technology products that run on Windows at the [Microsoft Accessibility](#) website.

If you use assistive technology, be sure to contact your assistive technology vendor before you upgrade your software or hardware to check for possible compatibility issues.

Documentation in alternative formats

If you have difficulty reading or handling printed materials, you can obtain the documentation for many Microsoft products in more accessible formats. You can view an index of accessible product documentation on the [Microsoft Accessibility](#) website.

In addition, you can obtain additional Microsoft publications from Learning Ally. Learning Ally distributes these documents to registered, eligible members of their distribution service. For information about the availability of Microsoft product documentation and books from Microsoft Press, contact:

Learning Ally
20 Roszel Road
Princeton, NJ 08540

Learning Ally website: <http://www.learningally.org>

Web addresses can change, so you might be unable to connect to the website or sites mentioned here.

Customer service for people with hearing impairments

If you are deaf or hard-of-hearing, complete access to Microsoft product and customer services is available through a text telephone (TTY/TDD) service:

- For customer service, contact Microsoft Sales Information Center at (800) 892-5234 between 6:30 AM and 5:30 PM Pacific Time, Monday through Friday, excluding holidays.
- For technical assistance in the United States, contact Microsoft Product Support Services at (800) 892-5234 between 6:00 AM and 6:00 PM Pacific Time, Monday through Friday, excluding holidays. In Canada, dial (905) 568-9641 between 8:00 AM and 8:00 PM Eastern Time, Monday through Friday, excluding holidays.

Microsoft Support Services are subject to the prices, terms, and conditions in place at the time the service is used.

For more information

For more information about how accessible technology for computers helps to improve the lives of people with disabilities, see the [Microsoft Accessibility](#) website.

TIP

To learn more about recent accessibility updates to Visual Studio, see the [Accessibility improvements in Visual Studio 2017 version 15.3](#) blog post.

See also

- [Resources for designing accessible applications](#)
- [Accessibility features of Visual Studio](#)
- [Accessibility for Visual Studio for Mac](#)

Popular keyboard shortcuts for Visual Studio

10/18/2019 • 2 minutes to read • [Edit Online](#)

You can access frequently used commands in Visual Studio by choosing these default keyboard shortcuts. For a complete list of commands that have default shortcuts, see [Default keyboard shortcuts](#).

The *Global* context means that the shortcut is applicable in any tool window in Visual Studio.

NOTE

You can [look up the shortcut](#) for any command by opening the **Options** dialog box, expanding the **Environment** node, and then choosing **Keyboard**.

Build

COMMANDS	KEYBOARD SHORTCUTS [CONTEXTS]
Build.BuildSolution	Ctrl+Shift+B [Global]
Build.Cancel	Ctrl+Break [Global]
Build.Compile	Ctrl+F7 [Global]
Build.RunCodeAnalysisonSolution	Alt+F11 [Global]

Debug

COMMANDS	KEYBOARD SHORTCUTS [CONTEXTS]
Debug.BreakatFunction	Ctrl+B [Global]
Debug.BreakAll	Ctrl+Alt+Break [Global]
Debug.DeleteAllBreakpoints	Ctrl+Shift+F9 [Global]
Debug.Exceptions	Ctrl+Alt+E [Global]
Debug.QuickWatch	Ctrl+Alt+Q [Global] or Shift+F9 [Global]
Debug.Restart	Ctrl+Shift+F5 [Global]
Debug.RunWithCursor	Ctrl+F10 [Global]
Debug.SetNextStatement	Ctrl+Shift+F10 [Global]
Debug.Start	F5 [Global]

COMMANDS	KEYBOARD SHORTCUTS [CONTEXTS]
Debug.StartWithoutDebugging	Ctrl+F5 [Global]
Debug.StepInto	F11 [Global]
Debug.StepOut	Shift+F11 [Global]
Debug.StepOver	F10 [Global]
Debug.StopDebugging	Shift+F5 [Global]
Debug.ToggleBreakpoint	F9 [Global]

Edit

COMMANDS	KEYBOARD SHORTCUTS [CONTEXTS]
Edit.BreakLine	Enter [Text Editor, Report Designer, Windows Forms Designer] or Shift+Enter [Text Editor]
Edit.CollapseToDefinitions	Ctrl+M, Ctrl+O [Text Editor]
Edit.CommentSelection	Ctrl+K, Ctrl+C [Text Editor]
Edit.CompleteWord	Alt+Right Arrow [Text Editor, Workflow Designer] or Ctrl+Spacebar [Text Editor, Workflow Designer] or Ctrl+K, W [Workflow Designer] or Ctrl+K, Ctrl+W [Workflow Designer]
Edit.Copy	Ctrl+C [Global] or Ctrl+Insert [Global]
Edit.Cut	Ctrl+X [Global] or Shift+Delete [Global]
Edit.Delete	Delete [Global, Team Explorer] or Shift+Delete [Sequence Diagram, UML Activity Diagram, Layer Diagram] or Ctrl+Delete [Class Diagram]
Edit.Find	Ctrl+F [Global]
Edit.FindAllReferences	Shift+F12 [Global]
Edit.FindinFiles	Ctrl+Shift+F [Global]

COMMANDS	KEYBOARD SHORTCUTS [CONTEXTS]
Edit.FindNext	F3 [Global]
Edit.FindNextSelected	Ctrl+F3 [Global]
Edit.FormatDocument	Ctrl+K, Ctrl+D [Text Editor]
Edit.FormatSelection	Ctrl+K, Ctrl+F [Text Editor]
Edit.GoTo	Ctrl+G [Global]
Edit.GoToDeclaration	Ctrl+F12 [Global]
Edit.GoToDefinition	F12
Edit.GoToFindCombo	Ctrl+D [Global]
Edit.GoToNextLocation	F8 [Global]
Edit.InsertSnippet	Ctrl+K, Ctrl+X [Global]
Edit.InsertTab	Tab [Report Designer, Windows Forms Designer, Text Editor]
Edit.LineCut	Ctrl+L [Text Editor]
Edit.LineDownExtendColumn	Shift+Alt+Down Arrow [Text Editor]
Edit.LineOpenAbove	Ctrl+Enter [Text Editor]
Edit.ListMembers	Ctrl+J [Text Editor, Workflow Designer] or Ctrl+K, Ctrl+L [Workflow Designer] or Ctrl+K, L [Workflow Designer]
Edit.NavigateTo	Ctrl+, [Global]
Edit.OpenFile	Ctrl+Shift+G [Global]
Edit.OvertypeMode	Insert [Text Editor]
Edit.ParameterInfo	Ctrl+Shift+Spacebar [Text Editor, Workflow Designer] or Ctrl+K, Ctrl+P [Workflow Designer] or Ctrl+K, P [Workflow Designer]
Edit.Paste	Ctrl+V [Global] or Shift+Insert [Global]
Edit.PeekDefinition	Alt+F12 [Text Editor]

COMMANDS	KEYBOARD SHORTCUTS [CONTEXTS]
Edit.Redo	Ctrl+Y [Global] or Shift+Alt+Backspace [Global] or Ctrl+Shift+Z [Global]
Edit.Replace	Ctrl+H [Global]
Edit.SelectAll	Ctrl+A [Global]
Edit.SelectCurrentWord	Ctrl+W [Text Editor]
Edit.SelectionCancel	Esc [Text Editor, Report Designer, Settings Designer, Windows Forms Designer, Managed Resources Editor]
Edit.SurroundWith	Ctrl+K, Ctrl+S [Global]
Edit.TabLeft	Shift+Tab [Text Editor, Report Designer, Windows Forms Editor]
Edit.ToggleAllOutlining	Ctrl+M, Ctrl+L [Text Editor]
Edit.ToggleBookmark	Ctrl+K, Ctrl+K [Text Editor]
Edit.ToggleCompletionMode	Ctrl+Alt+Space [Text Editor]
Edit.ToggleOutliningExpansion	Ctrl+M, Ctrl+M [Text Editor]
Edit.UncommentSelection	Ctrl+K, Ctrl+U [Text Editor]
Edit.Undo	Ctrl+Z [Global] or Alt+Backspace [Global]
Edit.WordDeleteToEnd	Ctrl+Delete [Text Editor]
Edit.WordDeleteToStart	Ctrl+Backspace [Text Editor]

File

COMMANDS	KEYBOARD SHORTCUTS [CONTEXTS]
File.Exit	Alt+F4 [Global]
File.NewFile	Ctrl+N [Global]
File.NewProject	Ctrl+Shift+N [Global]
File.NewWebSite	Shift+Alt+N [Global]
File.OpenFile	Ctrl+O [Global]

COMMANDS	KEYBOARD SHORTCUTS [CONTEXTS]
File.OpenProject	Ctrl+Shift+O [Global]
File.OpenWebSite	Shift+Alt+O [Global]
File.Rename	F2 [Team Explorer]
File.SaveAll	Ctrl+Shift+S [Global]
File.SaveSelectedItems	Ctrl+S [Global]
File.ViewinBrowser	Ctrl+Shift+W [Global]

Project

COMMANDS	KEYBOARD SHORTCUTS [CONTEXTS]
Project.AddExistingItem	Shift+Alt+A [Global]
Project.AddNewItem	Ctrl+Shift+A [Global]

Refactor

COMMAND	KEYBOARD SHORTCUT [CONTEXT]
Refactor.ExtractMethod	Ctrl+R, Ctrl+M [Global]

Tools

COMMAND	KEYBOARD SHORTCUT [CONTEXT]
Tools.AttachtoProcess	Ctrl+Alt+P [Global]

View

COMMANDS	KEYBOARD SHORTCUTS [CONTEXTS]
View.ClassView	Ctrl+Shift+C [Global]
View.EditLabel	F2 [Global]
View.ErrorList	Ctrl+\, Ctrl+E [Global] or Ctrl+\, E [Global]
View.NavigateBackward	Ctrl+- [Global]
View.NavigateForward	Ctrl+Shift+- [Global]

COMMANDS	KEYBOARD SHORTCUTS [CONTEXTS]
View.ObjectBrowser	Ctrl+Alt+J [Global]
View.Output	Ctrl+Alt+O [Global]
View.PropertiesWindow	F4
View.Refresh	F5 [Team Explorer]
View.ServerExplorer	Ctrl+Alt+S [Global]
View.ShowSmartTag	Ctrl+. [Global] or Shift+Alt+F10 [Global, HTML Editor Design View]
View.SolutionExplorer	Ctrl+Alt+L [Global]
View.TfsTeamExplorer	Ctrl+\, Ctrl+M [Global]
View.Toolbox	Ctrl+Alt+X [Global]
View.ViewCode	Enter [Class Diagram] or F7 [Settings Designer]
View.ViewDesigner	Shift+F7 [HTML Editor Source View]

Window

COMMANDS	KEYBOARD SHORTCUTS [CONTEXTS]
Window.ActivateDocumentWindow	Esc [Global]
Window.CloseDocumentWindow	Ctrl+F4 [Global]
Window.NextDocumentWindow	Ctrl+F6 [Global]
Window.NextDocumentWindowNav	Ctrl+Tab [Global]
Window.NextSplitPane	F6 [Global]

See also

- [Identify and customize keyboard shortcuts in Visual Studio](#)
- [All keyboard shortcuts in Visual Studio](#)

Default keyboard shortcuts in Visual Studio

10/18/2019 • 13 minutes to read • [Edit Online](#)

You can access a variety of [commands](#) and windows in Visual Studio by choosing the appropriate keyboard shortcut. This page lists the default command shortcuts for the **General** profile, which you might have chosen when you installed Visual Studio. No matter which profile you chose, you can [identify the shortcut](#) for a command by opening the **Options** dialog box, expanding the **Environment** node, and then choosing **Keyboard**. You can also customize your shortcuts by assigning a different shortcut to any given command.

For a list of common keyboard shortcuts and other productivity information, see:

- [Keyboard shortcuts for frequently used commands](#)
- [Keyboard tips](#)
- [Productivity tips](#).

For more information about accessibility in Visual Studio, see [Accessibility tips and tricks](#) and [How to: Use the keyboard exclusively](#).

Analyze	Edit	Project	Test
Architecture	Editor context menus	Project and Solution context menus	Test Explorer
Build	File	Refactor	Tools
Class View context menus	Help	Solution Explorer	View
Debug	Load test	Team	Window
Debugger context menus	Other context menus	Team Foundation context menus	Azure
Diagnostics hub			

Global shortcuts

These keyboard shortcuts are *global*, which means that you can use them when any Visual Studio window has focus.

Analyze

COMMANDS	KEYBOARD SHORTCUTS
Analyze.NavigateBackward	Shift+Alt+3
Analyze.NavigateForward	Shift+Alt+4

Architecture

COMMANDS	KEYBOARD SHORTCUTS
Architecture.NewDiagram	Ctrl+\, Ctrl+N

Build

COMMANDS	KEYBOARD SHORTCUTS
Build.BuildSelection	Ctrl+B (Visual Studio 2019)
Build.BuildSolution	Ctrl+Shift+B
Build.Cancel	Ctrl+Break
Build.Compile	Ctrl+F7
Build.RunCodeAnalysisonSolution	Alt+F11

Class View context menus

COMMANDS	KEYBOARD SHORTCUTS
ClassViewContextMenu.ClassViewMultiselectProjectreferencesItems.Properties	Alt+Enter

Debug

COMMANDS	KEYBOARD SHORTCUTS
Debug.ApplyCodeChanges	Alt+F10
Debug.Autos	Ctrl+Alt+V, A
Debug.BreakAll	Ctrl+Alt+Break
Debug.Breakpoints	Ctrl+Alt+B
Debug.CallStack	Ctrl+Alt+C
Debug.DeleteAllBreakpoints	Ctrl+Shift+F9
Debug.DiagnosticsHub.Launch	Alt+F2
Debug.Disassembly	Ctrl+Alt+D
Debug.DOMExplorer	Ctrl+Alt+V, D
Debug.EnableBreakpoint	Ctrl+F9
Debug.Exceptions	Ctrl+Alt+E
Debug.FunctionBreakpoint	Ctrl+K, B (Visual Studio 2019) Ctrl+B (Visual Studio 2017)

COMMANDS	KEYBOARD SHORTCUTS
Debug.GoToPreviousCallOrIntelliTraceEvent	Ctrl+Shift+F11
Debug.Graphics.StartDiagnostics	Alt+F5
Debug.Immediate	Ctrl+Alt+I
Debug.IntelliTraceCalls	Ctrl+Alt+Y, T
Debug.IntelliTraceEvents	Ctrl+Alt+Y, F
Debug.JavaScriptConsole	Ctrl+Alt+V, C
Debug.Locals	Ctrl+Alt+V, L
Debug.LocationToolbar.ProcessCombo	Ctrl+5
Debug.LocationToolbar.StackFrameCombo	Ctrl+7
Debug.LocationToolbar.ThreadCombo	Ctrl+6
Debug.LocationToolbar.ToggleCurrentThreadFlaggedState	Ctrl+8
Debug.LocationToolbar.ToggleFlaggedThreads	Ctrl+9
Debug.Memory1	Ctrl+Alt+M, 1
Debug.Memory2	Ctrl+Alt+M, 2
Debug.Memory3	Ctrl+Alt+M, 3
Debug.Memory4	Ctrl+Alt+M, 4
Debug.Modules	Ctrl+Alt+U
Debug.ParallelStacks	Ctrl+Shift+D, S
Debug.ParallelWatch1	Ctrl+Shift+D, 1
Debug.ParallelWatch2	Ctrl+Shift+D, 2
Debug.ParallelWatch3	Ctrl+Shift+D, 3
Debug.ParallelWatch4	Ctrl+Shift+D, 4
Debug.Processes	Ctrl+Alt+Z
Debug.QuickWatch	Shift+F9 or Ctrl+Alt+Q
Debug.RefreshWindowsapp	Ctrl+Shift+R

COMMANDS	KEYBOARD SHORTCUTS
Debug.Registers	Ctrl+Alt+G
Debug.Restart	Ctrl+Shift+F5
Debug.RunToCursor	Ctrl+F10
Debug.SetNextStatement	Ctrl+Shift+F10
Debug.ShowCallStackonCodeMap	Ctrl+Shift+`
Debug.ShowNextStatement	Alt+Num *
Debug.Start	F5
Debug.StartWindowsPhoneApplicationAnalysis	Alt+F1
Debug.StartWithoutDebugging	Ctrl+F5
Debug.StepInto	F11
Debug.StepIntoCurrentProcess	Ctrl+Alt+F11
Debug.StepIntoSpecific	Shift+Alt+F11
Debug.StepOut	Shift+F11
Debug.StepOutCurrentProcess	Ctrl+Shift+Alt+F11
Debug.StepOver	F10 (When debugging: Performs a step over action)
Debug.StepOver	F10 (When not debugging: Starts debugging and stops on the first line of user code)
Debug.StepOverCurrentProcess	Ctrl+Alt+F10
Debug.StopDebugging	Shift+F5
Debug.StopPerformanceAnalysis	Shift+Alt+F2
Debug.Tasks	Ctrl+Shift+D, K
Debug.Threads	Ctrl+Alt+H
Debug.ToggleBreakpoint	F9
Debug.ToggleDisassembly	Ctrl+F11
Debug.Watch1	Ctrl+Alt+W, 1
Debug.Watch2	Ctrl+Alt+W, 2

COMMANDS	KEYBOARD SHORTCUTS
Debug.Watch3	Ctrl+Alt+W, 3
Debug.Watch4	Ctrl+Alt+W, 4

Debugger context menus

COMMANDS	KEYBOARD SHORTCUTS
DebuggerContextMenus.BreakpointsWindow.Delete	Alt+F9, D
DebuggerContextMenus.BreakpointsWindow.GoToDisassembly	Alt+F9, A
DebuggerContextMenus.BreakpointsWindow.GoToSourceCode	Alt+F9, S

Diagnostics Hub

COMMAND	KEYBOARD SHORTCUT
DiagnosticsHub.StopCollection	Ctrl+Alt+F2

Edit

COMMANDS	KEYBOARD SHORTCUTS
Edit.Copy	Ctrl+C or Ctrl+Ins
Edit.Cut	Ctrl+X or Shift+Delete
Edit.CycleClipboardRing	Ctrl+Shift+V or Ctrl+Shift+Ins
Edit.Delete	Delete
Edit.Duplicate	Ctrl+D
Edit.Find	Ctrl+F
Edit.FindAllReferences	Shift+F12
Edit.FindinFiles	Ctrl+Shift+F

COMMANDS	KEYBOARD SHORTCUTS
Edit.FindNext	F3
Edit.FindNextSelected	Ctrl+F3
Edit.FindPrevious	Shift+F3
Edit.FindPreviousSelected	Ctrl+Shift+F3
Edit.GenerateMethod	Ctrl+K, Ctrl+M
Edit.GoTo	Ctrl+G
Edit.GoToAll	Ctrl+, or Ctrl+T
Edit.GoToDeclaration	Ctrl+F12
Edit.GoToDefinition	F12
Edit.GoToMember	Ctrl+1, Ctrl+M or Ctrl+1, M or Alt+\
Edit.GoToNextLocation	F8 (Next error in Error List or Output window)
Edit.GoToPrevLocation	Shift+F8 (Previous error in Error List or Output window)
Edit.InsertSnippet	Ctrl+K, Ctrl+X
Edit.MoveControlDown	Ctrl+ Down Arrow
Edit.MoveControlDownGrid	Down Arrow
Edit.MoveControlLeft	Ctrl+ Left Arrow
Edit.MoveControlLeftGrid	Left Arrow
Edit.MoveControlRight	Ctrl+ Right Arrow
Edit.MoveControlRightGrid	Right Arrow
Edit.MoveControlUp	Ctrl+ Up Arrow
Edit.MoveControlUpGrid	Up Arrow
Edit.NextBookmark	Ctrl+K, Ctrl+N
Edit.NextBookmarkInFolder	Ctrl+Shift+K, Ctrl+Shift+N
Edit.OpenFile	Ctrl+Shift+G (Opens the file name under the cursor)

COMMANDS	KEYBOARD SHORTCUTS
Edit.Paste	Ctrl+V or Shift+Ins
Edit.PreviousBookmark	Ctrl+K, Ctrl+P
Edit.PreviousBookmarkInFolder	Ctrl+Shift+K, Ctrl+Shift+P
Edit.QuickFindSymbol	Shift+Alt+F12
Edit.Redo	Ctrl+Y or Ctrl+Shift+Z or Shift+Alt+Backspace
Edit.RefreshRemoteReferences	Ctrl+Shift+J
Edit.Replace	Ctrl+H
Edit.ReplaceinFiles	Ctrl+Shift+H
Edit.SelectAll	Ctrl+A
Edit.SelectNextControl	Tab
Edit.SelectPreviousControl	Shift+Tab
Edit.ShowTileGrid	Enter
Edit.SizeControlDown	Ctrl+Shift+Down Arrow
Edit.SizeControlDownGrid	Shift+Down Arrow
Edit.SizeControlLeft	Ctrl+Shift+Left Arrow
Edit.SizeControlLeftGrid	Shift+Left Arrow
Edit.SizeControlRight	Ctrl+Shift+Right Arrow
Edit.SizeControlRightGrid	Shift+Right Arrow
Edit.SizeControlUp	Ctrl+Shift+Up Arrow
Edit.SizeControlUpGrid	Shift+Up Arrow

COMMANDS	KEYBOARD SHORTCUTS
Edit.StopSearch	Alt+F3, S
Edit.SurroundWith	Ctrl+K, Ctrl+S
Edit.Undo	Ctrl+Z or Alt+Backspace

Editor context menus

COMMANDS	KEYBOARD SHORTCUTS
EditorContextMenus.CodeWindow.Breakpoint.BreakpointEdtlabels	Alt+F9, L
EditorContextMenus.CodeWindow.CodeMap.ShowItem	Ctrl+`
EditorContextMenus.CodeWindow.Execute	Ctrl+Alt+F5
EditorContextMenus.CodeWindow.GoToView	Ctrl+M, Ctrl+G
EditorContextMenus.CodeWindow.ToggleHeaderCodeFile	Ctrl+K, Ctrl+O (letter 'O')
EditorContextMenus.CodeWindow.ViewCallHierarchy	Ctrl+K, Ctrl+T or Ctrl+K, T

File

COMMANDS	KEYBOARD SHORTCUTS
File.Exit	Alt+F4
File.NewFile	Ctrl+N
File.NewProject	Ctrl+Shift+N
File.NewWebSite	Shift+Alt+N
File.OpenFile	Ctrl+O (letter 'O')
File.OpenProject	Ctrl+Shift+O (letter 'O')
File.OpenWebSite	Shift+Alt+O (letter 'O')
File.Print	Ctrl+P
File.SaveAll	Ctrl+Shift+S

COMMANDS	KEYBOARD SHORTCUTS
File.SaveSelectedItems	Ctrl+S
File.ViewinBrowser	Ctrl+Shift+W

Help

COMMANDS	KEYBOARD SHORTCUTS
Help.AddandRemoveHelpContent	Ctrl+Alt+F1
Help.F1Help	F1
Help.ViewHelp	Ctrl+F1
Help.WindowHelp	Shift+F1

Load test

COMMAND	KEYBOARD SHORTCUT
LoadTest.JumpToCounterPane	Ctrl+R, Q

Other context menus

COMMAND	KEYBOARD SHORTCUT
OtherContextMenu.MicrosoftDataEntityDesignContext.AddNewDiagram	Insert

Project

COMMANDS	KEYBOARD SHORTCUTS
Project.AddExistingItem	Shift+Alt+A
Project.AddNewItem	Ctrl+Shift+A
Project.ClassWizard	Ctrl+Shift+X
Project.Override	Ctrl+Alt+Ins
Project.Previewchanges	Alt+; then Alt+C
Project.Publishselectedfiles	Alt+; then Alt+P
Project.Replaceselectedfilesfromserver	Alt+; then Alt+R

Project and solution context menus

COMMANDS	KEYBOARD SHORTCUTS
ProjectandSolutionContextMenus.Item.MoveDown	Alt+Down Arrow

COMMANDS**KEYBOARD SHORTCUTS**

ProjectandSolutionContextMenus.Item.MoveUp

Alt+Up Arrow**Refactor****COMMANDS****KEYBOARD SHORTCUTS**

Refactor.EncapsulateField

Ctrl+R, Ctrl+E

Refactor.ExtractInterface

Ctrl+R, Ctrl+I

Refactor.ExtractMethod

Ctrl+R, Ctrl+M

Refactor.RemoveParameters

Ctrl+R, Ctrl+V

Refactor.Rename

Ctrl+R, Ctrl+R

Refactor.ReorderParameters

Ctrl+R, Ctrl+O (letter 'O')**Solution Explorer****COMMANDS****KEYBOARD SHORTCUTS**

SolutionExplorer.OpenFilesFilter

Ctrl+[, O (letter 'O')

or

Ctrl+[, Ctrl+O (letter 'O')

SolutionExplorer.PendingChangesFilter

Ctrl+[, P

or

Ctrl+[, Ctrl+P

SolutionExplorer.SyncWithActiveDocument

Ctrl+[, S

or

Ctrl+[, Ctrl+S**Team****COMMANDS****KEYBOARD SHORTCUTS**

Team.Git.GoToGitBranches

Ctrl+0 (zero), Ctrl+N

or

Ctrl+0, N

COMMANDS	KEYBOARD SHORTCUTS
Team.Git.GoToGitChanges	Ctrl+0 (zero), Ctrl+G or Ctrl+0, G
Team.Git.GoToGitCommits	Ctrl+0 (zero), Ctrl+O (letter 'O') or Ctrl+0, O
Team.TeamExplorerSearch	Ctrl+*

Team Foundation context menus

COMMANDS	KEYBOARD SHORTCUTS
TeamFoundationContextMenus.Commands.GoToBuilds	Ctrl+0 (zero), Ctrl+B or Ctrl+0, B
TeamFoundationContextMenus.Commands.GoToConnect	Ctrl+0 (zero), Ctrl+C or Ctrl+0, C
TeamFoundationContextMenus.Commands.GoToDocuments	Ctrl+0 (zero), Ctrl+D or Ctrl+0, D
TeamFoundationContextMenus.Commands.GoToHome	Ctrl+0 (zero), Ctrl+H or Ctrl+0, H
TeamFoundationContextMenus.Commands.GoToMyWork	Ctrl+0 (zero), Ctrl+M or Ctrl+0, M
TeamFoundationContextMenus.Commands.GoToPendingChanges	Ctrl+0 (zero), Ctrl+P or Ctrl+0, P

COMMANDS	KEYBOARD SHORTCUTS
TeamFoundationContextMenus.Commands.GoToReports	Ctrl+0 (zero), Ctrl+R or Ctrl+0, R
TeamFoundationContextMenus.Commands.GoToSettings	Ctrl+0 (zero), Ctrl+S or Ctrl+0, S
TeamFoundationContextMenus.Commands.GoToWebAccess	Ctrl+0 (zero), Ctrl+A or Ctrl+0, A
TeamFoundationContextMenus.Commands.GoToWorkItems	Ctrl+0 (zero), Ctrl+W or Ctrl+0, W

Test

COMMANDS	KEYBOARD SHORTCUTS
Test.UseCodedUITestBuilder	Ctrl+\, Ctrl+C
Test.UseExistingActionRecording	Ctrl+\, Ctrl+A

Test Explorer

COMMANDS	KEYBOARD SHORTCUTS
TestExplorer.DebugAllTests	Ctrl+R, Ctrl+A
TestExplorer.DebugAllTestsInContext	Ctrl+R, Ctrl+T
TestExplorer.DebugLastRun	Ctrl+R, D
TestExplorer.RepeatLastRun	Ctrl+R, L
TestExplorer.RunAllTests	Ctrl+R, A
TestExplorer.RunAllTestsInContext	Ctrl+R, T
TestExplorer.ShowTestExplorer	Ctrl+E, T
LiveUnitTesting.OpenTab	Ctrl+E, L
Test.CodeCoverageResults	Ctrl+E, C

Tools

COMMANDS	KEYBOARD SHORTCUTS
Tools.AttachtoProcess	Ctrl+Alt+P
Tools.CodeSnippetsManager	Ctrl+K, Ctrl+B
Tools.ForceGC	Ctrl+Shift+Alt+F12, Ctrl+Shift+Alt+F12

View

COMMANDS	KEYBOARD SHORTCUTS
View.AllWindows	Shift+Alt+M
View.ArchitectureExplorer	Ctrl+\, Ctrl+R
View.Backward	Alt+Left Arrow (Functions differently from View.NavigateBackward in Text Editor)
View.BookmarkWindow	Ctrl+K, Ctrl+W
View.BrowseNext	Ctrl+Shift+1
View.BrowsePrevious	Ctrl+Shift+2
View.CallHierarchy	Ctrl+Alt+K
View.ClassView	Ctrl+Shift+C
View.ClassViewGoToSearchCombo	Ctrl+K, Ctrl+V
View.CodeDefinitionWindow	Ctrl+\, D or Ctrl+\, Ctrl+D
View.CommandWindow	Ctrl+Alt+A
View.DataSources	Shift+Alt+D
View.DocumentOutline	Ctrl+Alt+T
View.EditLabel	F2
View.ErrorList	Ctrl+\, E or Ctrl+\, Ctrl+E
View.F#Interactive	Ctrl+Alt+F

COMMANDS	KEYBOARD SHORTCUTS
View.FindSymbolResults	Ctrl+Alt+F12
View.Forward	Alt+Right Arrow (Functions differently from View.NavigateForward in Text Editor)
View.ForwardBrowseContext	Ctrl+Shift+7
View.FullScreen	Shift+Alt+Enter
View.NavigateBackward	Ctrl+-
View.NavigateForward	Ctrl+Shift+-
View.NextError	Ctrl+Shift+F12
View.Notifications	Ctrl+W, N or Ctrl+W, Ctrl+N
View.ObjectBrowser	Ctrl+Alt+J
View.ObjectBrowserGoToSearchCombo	Ctrl+K, Ctrl+R
View.Output	Ctrl+Alt+O (letter 'O')
View.PopBrowseContext	Ctrl+Shift+8 (C++ only)
View.PropertiesWindow	F4
View.PropertyPages	Shift+F4
View.ResourceView	Ctrl+Shift+E
View.ServerExplorer	Ctrl+Alt+S
View.ShowSmartTag	Shift+Alt+F10 or Ctrl+.
View.SolutionExplorer	Ctrl+Alt+L
View.SQLServerObjectExplorer	Ctrl+\, Ctrl+S
View.TaskList	Ctrl+\, T or Ctrl+\, Ctrl+T

COMMANDS	KEYBOARD SHORTCUTS
View.TfsTeamExplorer	Ctrl+\, Ctrl+M
View.Toolbox	Ctrl+Alt+X
View.UMLModelExplorer	Ctrl+\, Ctrl+U
View.ViewCode	F7
View.ViewDesigner	Shift+F7
View.WebBrowser	Ctrl+Alt+R
View.ZoomIn	Ctrl+Shift+.
View.ZoomOut	Ctrl+Shift+,
TestExplorer.ShowTestExplorer	Ctrl+E, T

Window

COMMANDS	KEYBOARD SHORTCUTS
Window.ActivateDocumentWindow	Esc
Window.AddTabtoSelection	Ctrl+Shift+Alt+Space
Window.CloseDocumentWindow	Ctrl+F4
Window.CloseToolWindow	Shift+Esc
Window.KeepTabOpen	Ctrl+Alt+Home
Window.MovetoNavigationBar	Ctrl+F2
Window.NextDocumentWindow	Ctrl+F6
Window.NextDocumentWindowNav	Ctrl+Tab
Window.NextPane	Alt+F6
Window.NextSplitPane	F6
Window.NextTab	Ctrl+Alt+PgDn or Ctrl+PgDn
Window.NextTabandAddtoSelection	Ctrl+Shift+Alt+PgDn
Window.NextToolWindowNav	Alt+F7

COMMANDS	KEYBOARD SHORTCUTS
Window.PreviousDocumentWindow	Ctrl+Shift+F6
Window.PreviousDocumentWindowNav	Ctrl+Shift+Tab
Window.PreviousPane	Shift+Alt+F6
Window.PreviousSplitPane	Shift+F6
Window.PreviousTab	Ctrl+Alt+PgUp or Ctrl+PgUp
Window.PreviousTabandAddtoSelection	Ctrl+Shift+Alt+PgUp
Window.PreviousToolWindowNav	Shift+Alt+F7
Window.QuickLaunch	Ctrl+Q
Window.QuickLaunchPreviousCategory	Ctrl+Shift+Q
Window.ShowDockMenu	Alt+-
Window.ShowEzMDIFileList	Ctrl+Alt+Down Arrow
Window.SolutionExplorerSearch	Ctrl+;
Window.WindowSearch	Alt+`

Azure

COMMANDS	KEYBOARD SHORTCUTS
WindowsAzure.RetryMobileServiceScriptOperation	Ctrl+Num *, Ctrl+R
WindowsAzure>ShowMobileServiceScriptErrorDetails	Ctrl+Num *, Ctrl+D

ADO.NET Entity Data Model Designer

COMMANDS	KEYBOARD SHORTCUTS
OtherContextMenus.MicrosoftDataEntityDesignContext.MoveProperties.Down	Alt+Down Arrow
OtherContextMenus.MicrosoftDataEntityDesignContext.MoveProperties.Down5	Alt+PgDn
OtherContextMenus.MicrosoftDataEntityDesignContext.MoveProperties.ToBottom	Alt+End

COMMANDS	KEYBOARD SHORTCUTS
OtherContextMenus.MicrosoftDataEntityDesignContext.MoveProperties.ToTop	Alt+Home
OtherContextMenus.MicrosoftDataEntityDesignContext.MoveProperties.Up	Alt+Up Arrow
OtherContextMenus.MicrosoftDataEntityDesignContext.MoveProperties.Up5	Alt+PgUp
OtherContextMenus.MicrosoftDataEntityDesignContext.Refactor.Rename	Ctrl+R, R
OtherContextMenus.MicrosoftDataEntityDesignContext.RemovefromDiagram	Shift+Del
View.EntityDataModelBrowser	Ctrl+1
View.EntityDataModelMappingDetails	Ctrl+2

Class diagram

COMMANDS	KEYBOARD SHORTCUTS
ClassDiagram.Collapse	Num -
ClassDiagram.Expand	Num +
Edit.Delete	Ctrl+Del
Edit.ExpandCollapseBaseTypeList	Shift+Alt+B
Edit.NavigateToLollipop	Shift+Alt+L
Edit.RemovefromDiagram	Delete
View.ViewCode	Enter

Coded UI Test Editor

COMMANDS	KEYBOARD SHORTCUTS
OtherContextMenus.UITestEditorContextMenu.CopyReferenceClipboard	Ctrl+C
OtherContextMenus.UITestEditorContextMenu.InsertDelayBefore	Ctrl+Alt+D
OtherContextMenus.UITestEditorContextMenu.LocateAll	Shift+Alt+L

COMMANDS	KEYBOARD SHORTCUTS
OtherContextMenus.UITestEditorContextMenu.LocatetheUIControl	Ctrl+Shift+L
OtherContextMenus.UITestEditorContextMenu.Movecode	Ctrl+Alt+C
OtherContextMenus.UITestEditorContextMenu.Splitintoanewmethod	Ctrl+Shift+T

DataSet Editor

COMMANDS	KEYBOARD SHORTCUTS
OtherContextMenus.ColumnContext.InsertColumn	Insert
OtherContextMenus.DbTableContext.Add.Column	Ctrl+L

Difference Viewer

COMMANDS	KEYBOARD SHORTCUTS
Diff.IgnoreTrimWhitespace	Ctrl+\, Ctrl+Spacebar
Diff.InlineView	Ctrl+\, Ctrl+1
Diff.LeftOnlyView	Ctrl+\, Ctrl+3
Diff.NextDifference	F8
Diff.PreviousDifference	Shift+F8
Diff.RightOnlyView	Ctrl+\, Ctrl+4
Diff.SideBySideView	Ctrl+\, Ctrl+2
Diff.SwitchBetweenLeftAndRight	Ctrl+\, Ctrl+Tab
Diff.SynchronizeViewToggle	Ctrl+\, Ctrl+Down Arrow
EditorContextMenus.CodeWindow.AddComment	Ctrl+Shift+K
EditorContextMenus.CodeWindow.EditLocalFile	Ctrl+Shift+P

DOM Explorer

COMMANDS	KEYBOARD SHORTCUTS
DOMExplorer.Refresh	F5
DOMExplorer.SelectElement	Ctrl+B

COMMANDS**KEYBOARD SHORTCUTS**

DOMExplorer.ShowLayout

Ctrl+Shift+I

F# interactive

COMMAND**KEYBOARD SHORTCUT**

OtherContextMenu.FSIConsoleContext.CancelInteractiveEvaluation

Ctrl+Break

Graph Document Editor

COMMANDS**KEYBOARD SHORTCUTS**

ArchitectureContextMenu.DirectedGraphContextMenu.Advanced.Add.AddNode

Insert

ArchitectureContextMenu.DirectedGraphContextMenu.Advanced.Select.BothDependencies

B

ArchitectureContextMenu.DirectedGraphContextMenu.Advanced.Select.IncomingDependencies

I

ArchitectureContextMenu.DirectedGraphContextMenu.Advanced.Select.OutgoingDependencies

O

ArchitectureContextMenu.DirectedGraphContextMenu.NewComment

Ctrl+Shift+K

or

Ctrl+E, C

ArchitectureContextMenu.DirectedGraphContextMenu.Remove

Delete

ArchitectureContextMenu.DirectedGraphContextMenu.Rename

F2

Graphics diagnostics

COMMANDS**KEYBOARD SHORTCUTS**

Debug.Graphics.CaptureFrame

None

Graphics.MovePixelSelectionDown

Shift+Alt+Down Arrow

Graphics.MovePixelSelectionLeft

Shift+Alt+Left Arrow

Graphics.MovePixelSelectionRight

Shift+Alt+Right Arrow

COMMANDS	KEYBOARD SHORTCUTS
Graphics.MovePixelSelectionUp	Shift+Alt+Up Arrow
Graphics.ZoomToActualSize	Shift+Alt+0 (zero)
Graphics.ZoomToFitInWindow	Shift+Alt+9
Graphics.ZoomIn	Shift+Alt+=
Graphics.ZoomOut	Shift+Alt+-

HTML Editor

COMMAND	KEYBOARD SHORTCUT
OtherContextMenu.HTMLContext.GoToController	Ctrl+M, Ctrl+G

HTML Editor Design View

COMMANDS	KEYBOARD SHORTCUTS
Edit.MoveControlDown	Ctrl+ Down Arrow
Edit.MoveControlUp	Ctrl+ Up Arrow
Format.Bold	Ctrl+B
Format.ConverttoHyperlink	Ctrl+L
Format.InsertBookmark	Ctrl+Shift+L
Format.Italic	Ctrl+I
Format.Underline	Ctrl+U
Project.AddContentPage	Ctrl+M, Ctrl+C
Table.ColumntotheLeft	Ctrl+Alt+ Left Arrow
Table.ColumntotheRight	Ctrl+Alt+ Right Arrow
Table.RowAbove	Ctrl+Alt+ Up Arrow
Table.RowBelow	Ctrl+Alt+ Down Arrow
View.ASP.NETNonvisualControls	Ctrl+Shift+N
View.EditMaster	Ctrl+M, Ctrl+M
View.NextView	Ctrl+ PgDn

COMMANDS	KEYBOARD SHORTCUTS
View.ShowSmartTag	Shift+Alt+F10
View.ViewMarkup	Shift+F7
Window.PreviousTab	Ctrl+PgUp

HTML Editor Source View

COMMANDS	KEYBOARD SHORTCUTS
OtherContextMenu.HTMLEContext.GoToController	Ctrl+M, Ctrl+G
View.NextView	Ctrl+PgDn
View.SynchronizeViews	Ctrl+Shift+Y
View.ViewDesigner	Shift+F7
Window.PreviousTab	Ctrl+PgUp

Layer diagram

COMMAND	KEYBOARD SHORTCUT
Edit.Delete	Shift+Delete

Managed Resources Editor

COMMANDS	KEYBOARD SHORTCUTS
Edit.EditCell	F2
Edit.Remove	Delete
Edit.RemoveRow	Ctrl+Delete
Edit.SelectionCancel	Escape
Resources.Audio	Ctrl+4
Resources.Files	Ctrl+5
Resources.Icons	Ctrl+3
Resources.Images	Ctrl+2
Resources.Other	Ctrl+6

COMMANDS	KEYBOARD SHORTCUTS
Resources.Strings	Ctrl+1

Merge Editor window

COMMANDS	KEYBOARD SHORTCUTS
TeamFoundationContextMenus.MergeContextMenu.SetFocusOnLeftWindow	Alt+1
TeamFoundationContextMenus.MergeContextMenu.SetFocusOnResultWindow	Alt+2
TeamFoundationContextMenus.MergeContextMenu.SetFocusOnRightWindow	Alt+3

Microsoft SQL Server Data Tools, Schema Compare

COMMANDS	KEYBOARD SHORTCUTS
SQL.SSDTSchemaCompareCompare	Shift+Alt+C
SQL.SSDTSchemaCompareGenerateScript	Shift+Alt+G
SQL.SSDTSchemaCompareNextChange	Shift+Alt+.
SQL.SSDTSchemaComparePreviousChange	Shift+Alt+,
SQL.SSDTSchemaCompareStop	Alt+Break
SQL.SSDTSchemaCompareWriteUpdates	Shift+Alt+U

Microsoft SQL Server Data Tools, Table Designer

COMMANDS	KEYBOARD SHORTCUTS
CommitAllEdits	Shift+Alt+U
SQL.ExpandWildcards	Ctrl+R, E or Ctrl+R, Ctrl+E
SQL.FullyqualifyNames	Ctrl+R, Q or Ctrl+R, Ctrl+Q

COMMANDS	KEYBOARD SHORTCUTS
SQL.MoveToSchema	Ctrl+R, M or Ctrl+R, Ctrl+M
SQL.Rename	F2 or Ctrl+R, R or Ctrl+R, Ctrl+R
ViewFileInScriptPanel	Shift+Alt+PgDn

Microsoft SQL Server Data Tools, T-SQL Editor

COMMANDS	KEYBOARD SHORTCUTS
CommitAllEdits	Shift+Alt+U
SQL.ExecuteWithDebugger	Alt+F5
SQL.ExpandWildcards	Ctrl+R, E or Ctrl+R, Ctrl+E
SQL.FullyqualifyNames	Ctrl+R, Q or Ctrl+R, Ctrl+Q
SQL.MoveToSchema	Ctrl+R, M or Ctrl+R, Ctrl+M
SQL.Rename	F2 or Ctrl+R, R or Ctrl+R, Ctrl+R
SQL.TSqlEditorCancelQuery	Alt+Break

COMMANDS	KEYBOARD SHORTCUTS
SQL.TSqlEditorExecuteQuery	Ctrl+Shift+E
SQL.TSqlEditorResultsAsFile	Ctrl+D, F
SQL.TSqlEditorResultsAsGrid	Ctrl+D, G
SQL.TSqlEditorResultsAsText	Ctrl+D, T
SQL.TSqlEditorShowEstimatedPlan	Ctrl+D, E
SQL.TSqlEditorToggleExecutionPlan	Ctrl+D, A
SQL.TSqlEditorToggleResultsPane	Ctrl+D, R
TSqlEditorCloneQuery	Ctrl+Alt+N
TSqlEditorDatabaseCombo	Shift+Alt+PgDn

Microsoft SQL Server Data Tools, T-SQL PDW Editor

COMMANDS	KEYBOARD SHORTCUTS
SQL.TSqlEditorCancelQuery	Alt+Break
SQL.TSqlEditorExecuteQuery	Ctrl+Shift+E
SQL.TSqlEditorResultsAsFile	Ctrl+D, F
SQL.TSqlEditorResultsAsGrid	Ctrl+D, G
SQL.TSqlEditorResultsAsText	Ctrl+D, T
SQL.TSqlEditorShowEstimatedPlan	Ctrl+D, E
SQL.TSqlEditorToggleExecutionPlan	Ctrl+D, A
SQL.TSqlEditorToggleResultsPane	Ctrl+D, R
TSqlEditorCloneQuery	Ctrl+Alt+N
TSqlEditorDatabaseCombo	Shift+Alt+PgDn

Page Inspector

COMMAND	KEYBOARD SHORTCUT
PageInspector.Minimize	F12

Query Designer

COMMANDS	KEYBOARD SHORTCUTS
QueryDesigner.CancelRetrievingData	Ctrl+T
QueryDesigner.Criteria	Ctrl+2
QueryDesigner.Diagram	Ctrl+1
QueryDesigner.ExecuteSQL	Ctrl+R
QueryDesigner.GotoRow	Ctrl+G
QueryDesigner.JoinMode	Ctrl+Shift+J
QueryDesigner.Results	Ctrl+4
QueryDesigner.SQL	Ctrl+3

Query results

COMMANDS	KEYBOARD SHORTCUTS
SQL.QueryResultsNewRow	Alt+End
SQL.QueryResultsRefresh	Shift+Alt+R
SQL.QueryResultsStop	Alt+Break

Report Designer

COMMANDS	KEYBOARD SHORTCUTS
Edit.BreakLine	Enter
Edit.CharLeft	Left Arrow
Edit.CharLeftExtend	Shift+Left Arrow
Edit.CharRight	Right Arrow
Edit.CharRightExtend	Shift+Right Arrow
Edit.InsertTab	Tab
Edit.LineDown	Down Arrow
Edit.LineDownExtend	Shift+Down Arrow
Edit.LineUp	Up Arrow

COMMANDS	KEYBOARD SHORTCUTS
Edit.LineUpExtend	Shift+Up Arrow
Edit.MoveControlDown	Ctrl+ Down Arrow
Edit.MoveControlLeft	Ctrl+ Left Arrow
Edit.MoveControlRight	Ctrl+ Right Arrow
Edit.MoveControlUp	Ctrl+ Up Arrow
Edit.SelectionCancel	Esc
Edit.SizeControlDown	Ctrl+Shift+ Down Arrow
Edit.SizeControlLeft	Ctrl+Shift+ Left Arrow
Edit.SizeControlRight	Ctrl+Shift+ Right Arrow
Edit.SizeControlUp	Ctrl+Shift+ Up Arrow
Edit.TabLeft	Shift+Tab
View.ReportData	Ctrl+Alt+D

Sequence diagram

COMMANDS	KEYBOARD SHORTCUTS
ArchitectureDesigner.Sequence.NavigateToCode	F12
Edit.Delete	Shift+Del

Settings Designer

COMMANDS	KEYBOARD SHORTCUTS
Edit.EditCell	F2
Edit.RemoveRow	Ctrl+ Delete
Edit.SelectionCancel	Esc
View.ViewCode	F7

Solution Explorer

COMMAND	KEYBOARD SHORTCUT
ClassViewContextMenu ClassViewProject View ViewInPage Inspector	Ctrl+K, Ctrl+G

Team Explorer

COMMAND	KEYBOARD SHORTCUT
Edit.Delete	Delete
File.Rename	F2
TeamFoundationContextMenu Commands.GoToTeamExplorerNavigation	Alt+Home
TeamFoundationContextMenu Commands.GoToTeamExplorerNextSectionContent	Alt+Down Arrow
TeamFoundationContextMenu Commands.GoToTeamExplorerPageContent	Alt+0 (zero)
TeamFoundationContextMenu Commands.GoToTeamExplorerPreviousSectionContent	Alt+Up Arrow
TeamFoundationContextMenu Commands.GoToTeamExplorerSection1Content	Alt+1
TeamFoundationContextMenu Commands.GoToTeamExplorerSection2Content	Alt+2
TeamFoundationContextMenu Commands.GoToTeamExplorerSection3Content	Alt+3
TeamFoundationContextMenu Commands.GoToTeamExplorerSection4Content	Alt+4
TeamFoundationContextMenu Commands.GoToTeamExplorerSection5Content	Alt+5
TeamFoundationContextMenu Commands.GoToTeamExplorerSection6Content	Alt+6
TeamFoundationContextMenu Commands.GoToTeamExplorerSection7Content	Alt+7
TeamFoundationContextMenu Commands.GoToTeamExplorerSection8Content	Alt+8
TeamFoundationContextMenu Commands.GoToTeamExplorerSection9Content	Alt+9
TeamFoundationContextMenu Commands.TeamExplorerNavigateBackward	Alt+Left Arrow

COMMAND	KEYBOARD SHORTCUT
TeamFoundationContextMenuCommands.TeamExplorerNavigateForward	Alt+Right Arrow
TeamFoundationContextMenuCommands.MyWorkPageInProgress.TfsContextMyWorkPageCreateCopyWI	Shift+Alt+C
TeamFoundationContextMenuCommands.MyWorkPageInProgress.TfsContextMyWorkPageNewLinkedWI	Shift+Alt+L
View.Refresh	F5

Test Explorer

COMMAND	KEYBOARD SHORTCUT
TestExplorer.OpenTest	F12

Text Editor

COMMANDS	KEYBOARD SHORTCUTS
Edit.BreakLine	Enter or Shift+Enter
Edit.CharLeft	Left Arrow
Edit.CharLeftExtend	Shift+Left Arrow
Edit.CharLeftExtendColumn	Shift+Alt+Left Arrow
Edit.CharRight	Right Arrow
Edit.CharRightExtend	Shift+Right Arrow
Edit.CharRightExtendColumn	Shift+Alt+Right Arrow
Edit.ClearBookmarks	Ctrl+K, Ctrl+L
Edit.CollapseAllOutlining	Ctrl+M, Ctrl+A
Edit.CollapseCurrentRegion	Ctrl+M, Ctrl+S
Edit.CollapseTag	Ctrl+M, Ctrl+T
Edit.CollapseToDefinitions	Ctrl+M, Ctrl+O (letter 'O')
Edit.ContractSelection	Shift+Alt+-

COMMANDS	KEYBOARD SHORTCUTS
Edit.CommentSelection	Ctrl+K, Ctrl+C
Edit.CompleteWord	Ctrl+Space or Alt+Right Arrow
Edit.CopyParameterTip	Ctrl+Shift+Alt+C
Edit.DecreaseFilterLevel	Alt+,
Edit.DeleteBackwards	Backspace or Shift+Bkspce
Edit.DeleteHorizontalWhiteSpace	Ctrl+K, Ctrl+\
Edit.DocumentEnd	Ctrl+End
Edit.DocumentEndExtend	Ctrl+Shift+End
Edit.DocumentStart	Ctrl+Home
Edit.DocumentStartExtend	Ctrl+Shift+Home
Edit.ExpandAllOutlining	Ctrl+M, Ctrl+X
Edit.ExpandCurrentRegion	Ctrl+M, Ctrl+E
Edit.ExpandSelection	Shift+Alt+=
Edit.ExpandSelectiontoContainingBlock	Shift+Alt+]
Edit.FormatDocument	Ctrl+K, Ctrl+D
Edit.FormatSelection	Ctrl+K, Ctrl+F
Edit.GotoAll	Ctrl+T or Ctrl+,
Edit.GotoBrace	Ctrl+]
Edit.GotoBraceExtend	Ctrl+Shift+]
Edit.GotoRecent	Ctrl+T,R

COMMANDS	KEYBOARD SHORTCUTS
Edit.GotoNextIssueinFile	Alt+PgDn
Edit.GotoPreviousIssueinFile	Alt+PgUp
Edit.HideSelection	Ctrl+M, Ctrl+H
Edit.IncreaseFilterLevel	Alt+.
Edit.IncrementalSearch	Ctrl+I
Edit.InsertCaretatAllMatching	Shift+Alt+;
Edit.InsertNextMatchingCaret	Shift+Alt+.
Edit.InsertTab	Tab
Edit.LineCut	Ctrl+L
Edit.LineDelete	Ctrl+Shift+L
Edit.LineDown	Down Arrow
Edit.LineDownExtend	Shift+Down Arrow
Edit.LineDownExtendColumn	Shift+Alt+Down Arrow
Edit.LineEnd	End
Edit.LineEndExtend	Shift+End
Edit.LineEndExtendColumn	Shift+Alt+End
Edit.LineOpenAbove	Ctrl+Enter
Edit.LineOpenBelow	Ctrl+Shift+Enter
Edit.LineStart	Home
Edit.LineStartExtend	Shift+Home
Edit.LineStartExtendColumn	Shift+Alt+Home
Edit.LineTranspose	Shift+Alt+T
Edit.LineUp	Up Arrow
Edit.LineUpExtend	Shift+Up Arrow
Edit.LineUpExtendColumn	Shift+Alt+Up Arrow

COMMANDS	KEYBOARD SHORTCUTS
Edit.ListMembers	Ctrl+J
Edit.MakeLowercase	Ctrl+U
Edit.MakeUppercase	Ctrl+Shift+U
Edit.MoveSelectedLinesDown	Alt+Down Arrow
Edit.MoveSelectedLinesUp	Alt+Up Arrow
Edit.NextHighlightedReference	Ctrl+Shift+Down Arrow
Edit.OvertypeMode	Insert
Edit.PageDown	PgDn
Edit.PageDownExtend	Shift+PgDn
Edit.PageUp	PgUp
Edit.PageUpExtend	Shift+PgUp
Edit.ParameterInfo	Ctrl+Shift+Spacebar
Edit.PasteParameterTip	Ctrl+Shift+Alt+P
Edit.PeekBackward	Ctrl+Alt+-
Edit.PeekDefinition	Alt+F12
Edit.PeekForward	Ctrl+Alt+=
Edit.PreviousHighlightedReference	Ctrl+Shift+Up Arrow
Edit.QuickInfo	Ctrl+K, Ctrl+I
Edit.ReverseIncrementalSearch	Ctrl+Shift+I
Edit.ScrollLineDown	Ctrl+Down Arrow
Edit.ScrollLineUp	Ctrl+Up Arrow
Edit.SelectCurrentWord	Ctrl+W
Edit.SelectionCancel	Escape
Edit.SelectToLastGoBack	Ctrl+=
Edit.ShowCodeLensMenu	Ctrl+K, Ctrl+`

COMMANDS	KEYBOARD SHORTCUTS
Edit.ShowNavigateMenu	Alt+`
Edit.StopHidingCurrent	Ctrl+M, Ctrl+U
Edit.StopOutlining	Ctrl+M, Ctrl+P
Edit.SwapAnchor	Ctrl+K, Ctrl+A
Edit.TabLeft	Shift+Tab
Edit.ToggleAllOutlining	Ctrl+M, Ctrl+L
Edit.ToggleBookmark	Ctrl+K, Ctrl+K
Edit.ToggleCompletionMode	Ctrl+Alt+Space
Edit.ToggleOutliningExpansion	Ctrl+M, Ctrl+M
Edit.ToggleTaskListShortcut	Ctrl+K, Ctrl+H
Edit.ToggleWordWrap	Ctrl+E, Ctrl+W
Edit.UncommentSelection	Ctrl+K, Ctrl+U
Edit.ViewBottom	Ctrl+PgDn
Edit.ViewBottomExtend	Ctrl+Shift+PgDn
Edit.ViewTop	Ctrl+PgUp
Edit.ViewTopExtend	Ctrl+Shift+PgUp
Edit.ViewWhiteSpace	Ctrl+R, Ctrl+W
Edit.WordDeleteToEnd	Ctrl+Delete
Edit.WordDeleteToStart	Ctrl+Backspace
Edit.WordNext	Ctrl+Right Arrow
Edit.WordNextExtend	Ctrl+Shift+Right Arrow
Edit.WordNextExtendColumn	Ctrl+Shift+Alt+Right Arrow
Edit.WordPrevious	Ctrl+Left Arrow
Edit.WordPreviousExtend	Ctrl+Shift+Left Arrow
Edit.WordPreviousExtendColumn	Ctrl+Shift+Alt+Left Arrow

COMMANDS	KEYBOARD SHORTCUTS
Edit.WordTranspose	Ctrl+Shift+T
EditorContextMenus.CodeWindow.ExecuteInInteractive	Alt+Enter
EditorContextMenus.CodeWindow.ExecuteLineInInteractive	Alt+'
OtherContextMenus.HTMLContext.ViewinPageInspector	Ctrl+K, Ctrl+G
TeamFoundationContextMenu.Annotate.TfsAnnotateMoveNextRegion	Alt+PgDn
TeamFoundationContextMenu.Annotate.TfsAnnotateMovePreviousRegion	Alt+PgUp

UML activity diagram

COMMAND	KEYBOARD SHORTCUT
Edit.Delete	Shift+Del

UML class diagram

COMMAND	KEYBOARD SHORTCUT
Edit.DeleteFromModel	Shift+Del

UML component diagram

COMMAND	KEYBOARD SHORTCUT
Edit.DeleteFromModel	Shift+Del

UML use case diagram

COMMAND	KEYBOARD SHORTCUT
Edit.DeleteFromModel	Shift+Del

VC Accelerator Editor

COMMANDS	KEYBOARD SHORTCUTS
Edit.NewAccelerator	Insert
Edit.NextKeyTyped	Ctrl+W

VC Dialog Editor

COMMANDS	KEYBOARD SHORTCUTS
Edit.MoveControlDown	Down Arrow
Edit.MoveControlLeft	Left Arrow
Edit.MoveControlRight	Right Arrow
Edit.MoveControlUp	Up Arrow
Edit.ScrollColumnLeft	Ctrl+Left Arrow
Edit.ScrollColumnRight	Ctrl+Right Arrow
Edit.ScrollLineDown	Ctrl+Down Arrow
Edit.ScrollLineUp	Ctrl+Up Arrow
Edit.SizeControlDown	Shift+Down Arrow
Edit.SizeControlLeft	Shift+Left Arrow
Edit.SizeControlRight	Shift+Right Arrow
Edit.SizeControlUp	Shift+Up Arrow
Format.AlignBottoms	Ctrl+Shift+Down Arrow
Format.AlignCenters	Shift+F9
Format.AlignLefts	Ctrl+Shift+Left Arrow
Format.AlignMiddles	F9
Format.AlignRights	Ctrl+Shift+Right Arrow
Format.AlignTops	Ctrl+Shift+Up Arrow
Format.ButtonBottom	Ctrl+B
Format.ButtonRight	Ctrl+R
Format.CenterHorizontal	Ctrl+Shift+F9
Format.CenterVertical	Ctrl+F9
Format.CheckMnemonics	Ctrl+M
Format.SizetoContent	Shift+F7

COMMANDS	KEYBOARD SHORTCUTS
Format.SpaceAcross	Alt+Right Arrow or Alt+Left Arrow
Format.SpaceDown	Alt+Up Arrow or Alt+Down Arrow
Format.TabOrder	Ctrl+D
Format.TestDialog	Ctrl+T
Format.ToggleGuides	Ctrl+G

VC Image Editor

COMMANDS	KEYBOARD SHORTCUTS
Image.AirbrushTool	Ctrl+A
Image.BrushTool	Ctrl+B
Image.CopyandOutlineSelection	Ctrl+Shift+U
Image.DrawOpaque	Ctrl+J
Image.EllipseTool	Alt+P
Image.EraseTool	Ctrl+Shift+I
Image.FilledEllipseTool	Ctrl+Shift+Alt+P
Image.FilledRectangleTool	Ctrl+Shift+Alt+R
Image.FilledRoundedRectangleTool	Ctrl+Shift+Alt+W
Image.FillTool	Ctrl+F
Image.FlipHorizontal	Ctrl+H
Image.FlipVertical	Shift+Alt+H
Image.LargerBrush	Ctrl+=
Image.LineTool	Ctrl+L
Image.MagnificationTool	Ctrl+M

COMMANDS	KEYBOARD SHORTCUTS
Image.Magnify	Ctrl+Shift+M
Image.NewImageType	Insert
Image.NextColor	Ctrl+] or Ctrl+Right Arrow
Image.NextRightColor	Ctrl+Shift+] or Ctrl+Shift+Right Arrow
Image.OutlinedEllipseTool	Shift+Alt+P
Image.OutlinedRectangleTool	Shift+Alt+R
Image.OutlinedRoundedRectangleTool	Shift+Alt+W
Image.PencilTool	Ctrl+I
Image.PreviousColor	Ctrl+[or Ctrl+Left Arrow
Image.PreviousRightColor	Ctrl+Shift+[or Ctrl+Shift+Left Arrow
Image.RectangleSelectionTool	Shift+Alt+S
Image.RectangleTool	Alt+R
Image.Rotate90Degrees	Ctrl+Shift+H
Image.RoundedRectangleTool	Alt+W
Image.ShowGrid	Ctrl+Alt+S
Image.ShowTileGrid	Ctrl+Shift+Alt+S
Image.SmallBrush	Ctrl+.
Image.SmallerBrush	Ctrl+-
Image.TextTool	Ctrl+T

COMMANDS	KEYBOARD SHORTCUTS
Image.UseSelectionasBrush	Ctrl+U
Image.ZoomIn	Ctrl+Shift+. or Ctrl+ Up Arrow
Image.ZoomOut	Ctrl+Shift+, or Ctrl+ Down Arrow

VC String Editor

COMMAND	KEYBOARD SHORTCUT
Edit.NewString	Insert

View Designer

COMMANDS	KEYBOARD SHORTCUTS
QueryDesigner.CancelRetrievingData	Ctrl+T
QueryDesigner.Criteria	Ctrl+2
QueryDesigner.Diagram	Ctrl+1
QueryDesigner.ExecuteSQL	Ctrl+R
QueryDesigner.GotoRow	Ctrl+G
QueryDesigner.JoinMode	Ctrl+Shift+J
QueryDesigner.Results	Ctrl+4
QueryDesigner.SQL	Ctrl+3

Visual Studio

COMMAND	KEYBOARD SHORTCUT
OtherContextMenus.ORDesignerContext.HideMethodsPane	Ctrl+1

Windows Forms Designer

COMMANDS	KEYBOARD SHORTCUTS
Edit.BreakLine	Enter
Edit.CharLeft	Left Arrow
Edit.CharLeftExtend	Shift+Left Arrow
Edit.CharRight	Right Arrow
Edit.CharRightExtend	Shift+Right Arrow
Edit.DocumentEnd	End
Edit.DocumentEndExtend	Shift+End
Edit.DocumentStart	Home
Edit.DocumentStartExtend	Shift+Home
Edit.InsertTab	Tab
Edit.LineDown	Down Arrow
Edit.LineDownExtend	Shift+Up Arrow
Edit.LineUp	Up Arrow
Edit.LineUpExtend	Shift+Down Arrow
Edit.MoveControlDown	Ctrl+Down Arrow
Edit.MoveControlLeft	Ctrl+Left Arrow
Edit.MoveControlRight	Ctrl+Right Arrow
Edit.MoveControlUp	Ctrl+Up Arrow
Edit.SelectionCancel	Escape
Edit.SizeControlDown	Ctrl+Shift+Down Arrow
Edit.SizeControlLeft	Ctrl+Shift+Left Arrow
Edit.SizeControlRight	Ctrl+Shift+Right Arrow
Edit.SizeControlUp	Ctrl+Shift+Up Arrow
Edit.TabLeft	Shift+Tab

Work Item Editor

COMMANDS	KEYBOARD SHORTCUTS
Edit.CreateCopyofWorkItem	Shift+Alt+C
Edit.RefreshWorkItem	F5
Team.NewLinkedWorkItem	Shift+Alt+L

Work Item Query View

COMMANDS	KEYBOARD SHORTCUTS
Edit.CreateCopyofWorkItem	Shift+Alt+C
Edit.Indent	Shift+Alt+Right Arrow
Edit.Outdent	Shift+Alt+Left Arrow
Team.NewLinkedWorkItem	Shift+Alt+L
Team.Refresh	F5
Window.Toggle	Shift+Alt+V

Work Item Results View

COMMANDS	KEYBOARD SHORTCUTS
Edit.CreateCopyofWorkItem	Shift+Alt+C
Edit.Indent	Shift+Alt+Right Arrow
Edit.Outdent	Shift+Alt+Left Arrow
Team.GotoNextWorkItem	Shift+Alt+N
Team.GotoPreviousWorkItem	Shift+Alt+P
Team.NewLinkedWorkItem	Shift+Alt+L
Team.Refresh	F5
Window.Toggle	Shift+Alt+V

Workflow Designer

COMMANDS	KEYBOARD SHORTCUTS

COMMANDS	KEYBOARD SHORTCUTS
Edit.CompleteWord	<p>Ctrl+K, W</p> <p>or</p> <p>Ctrl+K, Ctrl+W</p> <p>or</p> <p>Ctrl+Spacebar</p> <p>or</p> <p>Alt+Right Arrow</p>
Edit.DecreaseFilterLevel	Alt+,
Edit.IncreaseFilterLevel	Alt+.
Edit.ListMembers	<p>Ctrl+K, L</p> <p>or</p> <p>Ctrl+K, Ctrl+L</p> <p>or</p> <p>Ctrl+J</p>
Edit.ParameterInfo	<p>Ctrl+K, P</p> <p>or</p> <p>Ctrl+K, Ctrl+P</p> <p>or</p> <p>Ctrl+Shift+Spacebar</p>
Edit.QuickInfo	<p>Ctrl+K, I</p> <p>or</p> <p>Ctrl+K, Ctrl+I</p>
WorkflowDesigner.Collapse	<p>Ctrl+E, Ctrl+C</p> <p>or</p> <p>Ctrl+E, C</p>
WorkflowDesigner.CollapseAll	or
WorkflowDesigner.ConnectNodes	<p>Ctrl+E, Ctrl+F</p> <p>or</p> <p>Ctrl+E, F</p>

COMMANDS	KEYBOARD SHORTCUTS
WorkflowDesigner.CreateVariable	Ctrl+E, Ctrl+N or Ctrl+E, N
WorkflowDesigner.ExpandAll	Ctrl+E, Ctrl+X or Ctrl+E, X
WorkflowDesigner.ExpandInPlace	Ctrl+E, Ctrl+E or Ctrl+E, E
WorkflowDesigner.GoToParent	Ctrl+E, Ctrl+P or Ctrl+E, P
WorkflowDesigner.MoveFocus	Ctrl+E, Ctrl+M or Ctrl+E, M
WorkflowDesigner.NavigateThroughDesigner	Ctrl+Alt+F6
WorkflowDesigner.Restore	Ctrl+E, Ctrl+R or Ctrl+E, R
WorkflowDesigner.ShowHideArgumentDesigner	Ctrl+E, Ctrl+A or Ctrl+E, A
WorkflowDesigner.ShowHideImportsDesigner	Ctrl+E, Ctrl+I or Ctrl+E, I
WorkflowDesigner.ShowHideOverviewMap	Ctrl+E, Ctrl+O (letter 'O') or Ctrl+E, O

COMMANDS	KEYBOARD SHORTCUTS
WorkflowDesigner.ShowHideVariableDesigner	Ctrl+E, Ctrl+V or Ctrl+E, V
WorkflowDesigner.ToggleSelection	Ctrl+E, Ctrl+S or Ctrl+E, S
WorkflowDesigner.ZoomIn	Ctrl+Num +
WorkflowDesigner.ZoomOut	Ctrl+Num -

XAML UI Designer

COMMANDS	KEYBOARD SHORTCUTS
Design.FitAll	Ctrl+0 (zero)
Design>ShowHandles	F9
Design.ZoomIn	Ctrl+Alt+=
Design.ZoomOut	Ctrl+Alt+-
Designer options	Ctrl+Shift+;
Format>EditText	F2
Format>ResetLayout.All	Ctrl+Shift+R
Run project code	Ctrl+F9
Timeline>Hide (Blend only)	Ctrl+H
Timeline>Lock (Blend only)	Ctrl+L
Timeline>Show (Blend only)	Ctrl+Shift+H
Timeline>Unlock (Blend only)	Ctrl+Shift+L
View>EdgeLeftMoveLeft	Ctrl+Shift+,
View>EdgeLeftMoveRight	Ctrl+Shift+.
View>EdgeRightMoveLeft	Ctrl+Shift+Alt+,
View>EdgeRightMoveRight	Ctrl+Shift+Alt+.

COMMANDS	KEYBOARD SHORTCUTS
View.ShowPropertyMarkerMenu	Ctrl+Spacebar

XML (Text) Editor

COMMANDS	KEYBOARD SHORTCUTS
XML.StartXSLTDebugging	Alt+F5
XML.StartXSLTWithoutDebugging	Ctrl+Alt+F5

XML Schema Designer

COMMANDS	KEYBOARD SHORTCUTS
GraphView.BottomtoTop	Alt+Up Arrow
GraphView.LefttoRight	Alt+ Right Arrow
GraphView.RighttoLeft	Alt+ Left Arrow
GraphView.TopBottom	Alt+ Down Arrow
OtherContextMenus.GraphView.RemovefromWorkspace	Delete
XsdDesigner.ShowContentModelView	Ctrl+2
XsdDesigner.ShowGraphView	Ctrl+3
XsdDesigner.ShowStartView	Ctrl+1

See also

- [Visual Studio commands](#)

Visual Studio commands

10/18/2019 • 3 minutes to read • [Edit Online](#)

You can enter Visual Studio commands in the **Command** window, **Immediate** window, or **Find/Command** box. In each case, the greater than sign (>) indicates that a command, rather than a search or debug operation, follows.

You can find a complete list of commands and their syntax on the **Keyboard** page in **Tools > Options > Environment**.

In localized versions of the IDE, command names can be entered either in the native language of the IDE or in English. For example, you can type either `File.NewFile` or `Fichier.NouveauFichier` in the French IDE to execute the same command.

Many commands have aliases. For a list of command aliases, see [Command aliases](#). For command keyboard shortcuts, see [Default keyboard shortcuts in Visual Studio](#).

Escape character

The escape character for Visual Studio commands is a caret (^). The escape character means that the character immediately following it is interpreted literally rather than as a control character. This can be used to embed straight quotation marks ("), spaces, leading slashes, carets, or any other literal characters in a parameter or switch value, with the exception of switch names. For example:

```
>Edit.Find ^^t /regex
```

A caret functions the same whether it is inside or outside quotation marks. If a caret is the last character on the line, it's ignored.

Commands with arguments

The following commands take arguments or switches:

COMMAND NAME	DESCRIPTION
Add Existing Item	Adds an existing file to the current solution and opens it.
Add Existing Project	Adds an existing project to the current solution.
Add New Item	Adds a new solution item, such as an .htm, .css, .txt, or frameset to the current solution and opens it.
Alias	Creates a new alias for a complete command, complete command and arguments, or even another alias.
Evaluate Statement	Evaluates and displays the given statement.
Find	Searches files using a subset of the options available on the Find and Replace control.

COMMAND NAME	DESCRIPTION
Find in Files	Searches files using a subset of the options available on the Find in Files .
Go To	Moves the cursor to the specified line.
List Call Stack	Displays the current call stack.
List Disassembly	Begins the debug process and allows you to specify how errors are handled.
List Memory	Displays the contents of the specified range of memory.
List Modules	Lists the modules for the current process.
List Registers	Displays a list of registers.
List Source	Displays the specified lines of source code.
List Threads	Displays a list of the threads in the current program.
Log Command Window Output	Copies all input and output from the Command window into a file.
New File	Creates a new file and adds it to the currently selected project.
Open File	Opens an existing file and allows you to specify an editor.
Open Project	Opens an existing project and allows you to add the project to the current solution.
Print	Evaluates the expression and displays the results or the specified text.
Quick Watch Command	Displays the selected or specified text in the Expression field of the Quick Watch dialog box.
Replace	Replaces text in files using a subset of the options available on the Find and Replace control.
Replace in Files	Replaces text in files using a subset of the options available in the Replace in Files .
Set Current Stack Frame	Allows you to view a particular stack frame.
Set Current Thread	Allows you to view a particular thread.
Set Radix	Determines the number of bytes to view.
Shell	Launches programs from within Visual Studio as though the command has been executed from the command prompt.

COMMAND NAME	DESCRIPTION
ShowWebBrowser Command	Displays the URL you specify in a web browser window either within the integrated development environment (IDE) or external to the IDE.
Start	Begins the debug process and allows you to specify how errors are handled.
Path	Sets the list of directories for the debugger to search for symbols.
Toggle Breakpoint	Turns the breakpoint either on or off, depending on its current state, at the current location in the file.
Watch Command	Creates and opens a specified instance of a Watch window.

See also

- [Command window](#)
- [Find/Command box](#)
- [Visual Studio command aliases](#)

Visual Studio Command Aliases

10/21/2019 • 3 minutes to read • [Edit Online](#)

Command aliases let you type fewer characters when you want to execute a command. You enter aliases into the **Find/Command** box or **Command** window. For example, instead of entering `>File.OpenFile` to display the **Open File** dialog box, you can use the pre-defined alias `>of`.

Type `alias` in the **Command** window to display a list of the current aliases and their definitions. Type `>cls` to clear the contents of the **Command** window. If you want to see an alias for a specific command, type `alias <command name>`.

You can easily create your own alias for one of the Visual Studio commands (with or without arguments). For example, the syntax for aliasing `File.NewFile MyFile.txt` is `alias MyAlias File.NewFile MyFile.txt`. You can delete one of your aliases with `alias <alias name> /delete`.

The table below contains a list of the pre-defined Visual Studio command aliases. Some command names have more than one pre-defined alias. Click the links for the command names below to display detailed topics that explain the correct syntax, arguments, and switches for those commands.

COMMAND NAME	ALIAS	COMPLETE NAME
Print Command	?	Debug.Print
Quick Watch Command	??	Debug.Quickwatch
Add New Project	AddProj	File.AddNewProject
Alias Command	Alias	Tools.Alias
Autos window	Autos	Debug.Autos
Breakpoints window	bl	Debug.Breakpoints
Toggle Breakpoint	bp	Debug.ToggleBreakPoint
Call Stack window	CallStack	Debug.CallStack
Clear Bookmarks	ClearBook	Edit.ClearBookmarks
Close	Close	File.Close
Close All Documents	CloseAll	Window.CloseAllDocuments
Clear All	cls	Edit.ClearAll
Command mode	cmd	View.CommandWindow
View Code	code	View.ViewCode
List Memory Command	d	Debug.ListMemory

COMMAND NAME	ALIAS	COMPLETE NAME
List Memory Command as ANSI	da	Debug.ListMemory /Ansi
List Memory Command One-Byte format	db	Debug.ListMemory /Format:OneByte
List Memory Command as ANSI with Four-Byte format	dc	Debug.ListMemory /Format:FourBytes /Ansi
List Memory Command Four-Byte format	dd	Debug.ListMemory /Format:FourBytes
Delete to BOL	DelBOL	Edit.DeleteToBOL
Delete to EOL	DelEOL	Edit.DeleteToEOL
Delete Horizontal Whitespace	DelHSp	Edit.DeleteHorizontalWhitespace
View Designer	designer	View.ViewDesigner
List Memory Command Float format	df	Debug.ListMemory/Format:Float
Disassembly window	disasm	Debug.Disassembly
List Memory Command Eight-Byte format	dq	Debug.ListMemory /Format:EightBytes
List Memory Command as Unicode	du	Debug.ListMemory /Unicode
Evaluate Statement Command	eval	Debug.EvaluateStatement
Exit	Exit	File.Exit
Format Selection	format	Edit.FormatSelection
Full Screen	FullScreen	View.FullScreen
Start Command	g	Debug.Start
Go To Command	GotoLn	Edit.GoTo
Go to Brace	GotoBrace	Edit.GotoBrace
F1 Help	Help	Help.F1Help
Immediate Mode	immed	Tools.ImmediateMode
Insert File as Text	InsertFile	Edit.InsertFileAsText
List Call Stack Command	kb	Debug.ListCallStack
Make Lower Case	Lcase	Edit.MakeLowercase

COMMAND NAME	ALIAS	COMPLETE NAME
Cut Line	LineCut	Edit.LineCut
Delete Line	LineDel	Edit.LineDelete
List Members	ListMembers	Edit.ListMembers
Locals window	Locals	Debug.Locals
Log Command Window Output Command	Log	Tools.LogCommandWindowOutput
Command Window Mark Mode	mark	Tools.CommandWindowMarkMode
Memory window	Memory Memory1	Debug.Memory1
Memory Window 2	Memory2	Debug.Memory2
Memory Window 3	Memory3	Debug.Memory3
Memory Window 4	Memory4	Debug.Memory4
Set Radix Command	n	Debug.SetRadix
ShowWebBrowser Command	nav navigate	View.ShowWebBrowser
Next Bookmark	NextBook	Edit.NextBookmark
New File Command	nf	File.NewFile
New Project	np NewProj	File.NewProject
Open File Command	of Open	File.OpenFile
Open Project Command	op	File.OpenProject
Collapse to Definitions/Stop Outlining	OutlineDefs StopOutlining	Edit.CollapseToDefinitions
Step Over	p	Debug.StepOver
Parameter Information	ParamInfo	Edit.ParameterInfo
Step Out	pr	Debug.StepOut
Previous Bookmark	PrevBook	Edit.PreviousBookmark
Print File	print	File.Print
Properties Window	props	View.PropertiesWindow
Stop	q	Debug.StopDebugging

COMMAND NAME	ALIAS	COMPLETE NAME
Redo	redo	Edit.Redo
Registers window	registers	Debug.Registers
Run to Cursor	rtc	Debug.RunWithCursor
Save Selected Items	save	File.SaveSelectedItems
Save All	SaveAll	File.WriteAll
Save As	SaveAs	File.SaveSelectedItemsAs
Shell Command	shell	Tools.Shell
Stop Find In Files	StopFind	Edit.FindInFiles /stop
Swap Anchor	SwapAnchor	Edit.SwapAnchor
Step Into	t	Debug.StepInto
Tabify Selection	tabify	Edit.TabifySelection
Tasklist window	TaskList	View.TaskList
Threads window	Threads	Debug.Threads
Tile Horizontally	TileH	Window.TileHorizontally
Tile Vertically	TileV	Window.TileVertically
Toggle Bookmark	ToggleBook	Edit.ToggleBookmark
Toolbox window	toolbox	View.Toolbox
List Disassembly Command	u	Debug.ListDisassembly
Make Uppercase	Ucase	Edit.MakeUppercase
Undo	undo	Edit.Undo
Untabify Selection	Untabify	Edit.UntabifySelection
Watch window	Watch	Debug.WatchN
Toggle Word Wrap	WordWrap	Edit.ToggleWordWrap
List Processes		Debug.ListProcesses
List Threads Command	~ ~*k ~*kb	Debug.ListThreads Debug.ListThreads /AllThreads

See also

- [Visual Studio Commands](#)
- [Command Window](#)
- [Find/Command Box](#)

Add Existing Item Command

10/21/2019 • 2 minutes to read • [Edit Online](#)

Adds an existing file to the current solution and opens it.

Syntax

```
File.AddExistingItem filename [/e:editorname]
```

Arguments

`filename`

Required. The full path and file name, with extension, of the item to add to the current solution. If the file path or file name contains spaces, enclose the entire path in quotation marks.

Switches

`/e: editorname`

Optional. Name of the editor in which the file will be opened. If the argument is specified but no editor name is supplied, the **Open With** dialog box appears.

The `/e: editorname` argument syntax uses the editor names as they appear in the **Open With Dialog Box**, enclosed in quotation marks. For example, to open a style sheet in the source code editor, you would enter the following for the `/e: editorname` argument.

```
/e:"Source Code (text) Editor"
```

Remarks

Autocompletion tries to locate the correct path and file name as you type.

Example

This example adds the file, Form1.frm, to the current solution.

```
>File.AddExistingItem "C:\public\solution files\Form1.frm"
```

See also

- [Visual Studio Commands](#)
- [Command Window](#)
- [Find/Command Box](#)
- [Visual Studio Command Aliases](#)

Add Existing Project Command

10/21/2019 • 2 minutes to read • [Edit Online](#)

Adds an existing project to the current solution.

Syntax

```
File.AddExistingProject filename
```

Arguments

`filename`

Optional. The full path and project name, with extension, of the project to add to the solution.

If the `filename` argument includes spaces, it must be enclosed in quotation marks.

If no filename is specified, the command will open the file dialog so that user can pick a project.

Remarks

Auto completion tries to locate the correct path and file name as you type.

Example

This example adds the Visual Basic project, TestProject1, to the current solution.

```
>File.AddExistingProject "c:\visual studio projects\TestProject1.vbproj"
```

See also

- [Visual Studio Commands](#)
- [Command Window](#)
- [Find/Command Box](#)
- [Visual Studio Command Aliases](#)

Add New Item Command

10/21/2019 • 2 minutes to read • [Edit Online](#)

Adds a new solution item, such as an .htm, .css, .txt, or frameset to the current solution and opens it.

Syntax

```
File.AddNewItem [filename] [/t:templatename] [/e:editorname]
```

Arguments

`filename`

Optional. The path and file name of the item to add to the solution.

Switches

`/t: templatename`

Optional. Specifies the type of file to be created. If no template name is given, a text file is created by default.

The `/t: templatename` argument syntax mirrors the information found in the **Add New Solution Item** dialog box.

You must enter the complete category followed by the file type, separating the category name from the file type by a backslash (`\`) and enclosing the entire string in quotation marks.

For example, to create a new text file, you would enter the following for the `/t: templatename` argument.

```
/t:"General\Style Sheet"
```

`/e: editorname`

Optional. The name of the editor in which the file will be opened. If the argument is specified but no editor name is supplied, the **Open With** dialog box appears.

The `/e: editorname` argument syntax uses the editor names as they appear in the **Open With Dialog Box**, enclosed in quotation marks.

For example, to open a style sheet in the source code editor, you would enter the following for the `/e: editorname` argument.

```
/e:"Source Code (text) Editor"
```

Example

This example adds a new solution item, MyHTMLpg, to the current solution.

```
>File.AddNewItem MyHTMLpg /t:"General\HTML Page"
```

See also

- [Visual Studio Commands](#)
- [Command Window](#)
- [Find/Command Box](#)
- [Visual Studio Command Aliases](#)

Alias Command

10/21/2019 • 2 minutes to read • [Edit Online](#)

Creates a new alias for a complete command, complete command and arguments, or another alias.

TIP

Typing `>alias` without any arguments displays the current list of aliases and their definitions.

Syntax

```
Tools.Alias [/delete] [/reset] [aliasname] [aliasstring]
```

Arguments

`aliasname`

Optional. The name for the new alias. If no value is supplied for `aliasname`, a list of the current aliases and their definitions appears.

`aliasstring`

Optional. The complete command name or existing alias and any parameters that you want to create as an alias. If no value is supplied for `aliasstring`, the alias name and alias string for the specified alias displays.

Switches

/delete or /del or /d

Optional. Deletes the specified alias, removing it from autocompletion.

/reset

Optional. Resets the list of pre-defined aliases to its original settings. That is, it restores all pre-defined aliases and removes all user-defined aliases.

Remarks

Since aliases represent commands, they must be located at the beginning of the command line.

When issuing this command, you should include the switches immediately after the command, not after the aliases, otherwise the switch itself will be included as part of the alias string.

The `/reset` switch asks for a confirmation before the aliases are restored. There is no short form of `/reset`.

Examples

This example creates a new alias, `upper`, for the complete command `Edit.MakeUpperCase`.

```
>Tools.Alias upper Edit.MakeUpperCase
```

This example deletes the alias, `upper`.

```
>Tools.alias /delete upper
```

This example displays a list of all current aliases and definitions.

```
>Tools.Alias
```

See also

- [Visual Studio Commands](#)
- [Command Window](#)
- [Find/Command Box](#)
- [Visual Studio Command Aliases](#)

Evaluate Statement command

10/18/2019 • 2 minutes to read • [Edit Online](#)

Evaluates and displays the given statement.

Syntax

```
>Debug.EvaluateStatement text
```

Arguments

`text`

Required. The statement to evaluate.

Example

```
>Debug.EvaluateStatement args.Length
```

See also

- [Print Command](#)
- [Visual Studio Commands](#)
- [Command Window](#)
- [Find/Command Box](#)
- [Visual Studio Command Aliases](#)

Find Command

10/21/2019 • 2 minutes to read • [Edit Online](#)

Searches files using a subset of the options available on the **Find in Files** tab of the **Find and Replace** window.

Syntax

```
Edit.Find findwhat [/case] [/doc | /proc | /open | /sel]  
[/markall] [/options] [/reset] [/up] [/wild | /regex] [/word]
```

Arguments

`findwhat` Required. The text to match.

Switches

/case or /c

Optional. Matches occur only if the uppercase and lowercase characters exactly match those specified in the `findwhat` argument.

/doc or /d

Optional. Searches the current document only. Specify only one of the available search scopes, `/doc`, `/proc`, `/open`, or `/sel`.

/markall or /m

Optional. Places a graphic on each line that contains a search match within the current document.

/open or /o

Optional. Searches all open documents as if they were one document. Specify only one of the available search scopes, `/doc`, `/proc`, `/open`, or `/sel`.

/options or /t

Optional. Displays a list of the current find option settings and does not perform a search.

/proc or /p

Optional. Searches the current procedure only. Specify only one of the available search scopes, `/doc`, `/proc`, `/open`, or `/sel`.

/reset or /e

Optional. Returns the find options to their default settings and does not perform a search.

/sel or /s

Optional. Searches the current selection only. Specify only one of the available search scopes, `/doc`, `/proc`, `/open`, or `/sel`.

/up or /u

Optional. Searches from the current location in the file toward the beginning of the file. By default, searches begin at the current location in the file and searches towards the end of the file.

/regex or /r

Optional. Uses pre-defined special characters in the `findwhat` argument as notations that represent patterns of

text rather than the literal characters. For a complete list of regular expression characters, see [Regular Expressions](#).

/wild or /l

Optional. Uses pre-defined special characters in the `findwhat` argument as notations to represent a character or sequence of characters.

/word or /w

Optional. Searches only for whole words.

Example

This example performs a case-sensitive search for the word "somestring" in the currently selected section of code.

```
>Edit.Find somestring /sel /case
```

See also

- [Command Window](#)
- [Find/Command Box](#)
- [Visual Studio Commands](#)
- [Visual Studio Command Aliases](#)

Find in Files Command

10/21/2019 • 2 minutes to read • [Edit Online](#)

Search files using a subset of the options available on the **Find in Files** tab of the **Find and Replace** window.

Syntax

```
Edit.FindinFiles findwhat [/case] [/ext:extensions]
[/lookin:searchpath] [/names] [/options] [/reset] [/stop] [/sub]
[/text2] [/wild|/regex] [/word]
```

Arguments

`findwhat`

Required. The text to match.

Switches

/case or /c

Optional. Matches occur only if the uppercase and lowercase characters exactly match those specified in the `findwhat` argument.

/ext: `extensions`

Optional. Specifies the file extensions for the files to be searched. If not specified, the previous extension is used if one was previously entered.

/lookin: `searchpath`

Optional. Directory to search. If the path contains spaces, enclose the entire path in quotation marks.

/names or /n

Optional. Displays a list of file names that contain matches.

/options or /t

Optional. Displays a list of the current find option settings and does not perform a search.

/regex or /r

Optional. Uses pre-defined special characters in the `findwhat` argument as notations that represent patterns of text rather than the literal characters. For a complete list of regular expression characters, see [Regular Expressions](#).

/reset or /e

Optional. Returns the find options to their default settings and does not perform a search.

/stop

Optional. Halts the current search operation if one is in progress. Search ignores all other arguments when `/stop` has been specified. For example, to stop the current search you would enter the following:

```
>Edit.FindinFiles /stop
```

/sub or /s

Optional. Searches the subfolders within the directory specified in the `/lookin: searchpath` argument.

/text2 or /2

Optional. Displays the results of the search in the Find Results 2 window.

/wild or /l

Optional. Uses pre-defined special characters in the `findwhat` argument as notations to represent a character or sequence of characters.

/word or /w

Optional. Searches only for whole words.

Example

This example searches for btnCancel in all .cls files located in the folder "My Visual Studio Projects" and displays the match information in the Find Results 2 Window.

```
>Edit.FindinFiles btnCancel /lookin:"c:/My Visual Studio Projects" /ext:*.cls /text2
```

See also

- [Find in Files](#)
- [Command Window](#)
- [Find/Command Box](#)
- [Visual Studio Commands](#)
- [Visual Studio Command Aliases](#)

Go To Command

10/21/2019 • 2 minutes to read • [Edit Online](#)

Moves the cursor to the specified line.

Syntax

```
Edit.GoTo [linenumber]
```

Arguments

linenumber

Optional. An integer representing the number of the line to go to.

Remarks

The line numbering begins at one. If the value of `linenumber` is less than one, the first line displays. If the value of `linenumber` is greater than the number of the last line, the last line displays.

If a value for `linenumber` is not specified, the **Go To Line** dialog box displays.

The alias for this command is GoToLn.

Example

```
>Edit.GoTo 125
```

See also

- [Visual Studio Commands](#)
- [Command Window](#)
- [Find/Command Box](#)
- [Visual Studio Command Aliases](#)

Import and Export Settings command

10/18/2019 • 2 minutes to read • [Edit Online](#)

Imports, exports, or resets Visual Studio settings.

Syntax

```
Tools.ImportandExportSettings [/export:filename | /import:filename | /reset]
```

Switches

/export: `filename`

Optional. Exports the current settings to the specified file.

/import: `filename`

Optional. Imports the settings in the specified file.

/reset

Optional. Resets the current settings.

Remarks

Running this command with no switches opens the **Import and Export Settings** wizard. For more information, see [Synchronize your settings](#) and [Environment settings](#).

Example

The following command exports the current settings to the file `MyFile.vssettings`:

```
Tools.ImportandExportSettings /export:"c:\Files\MyFile.vssettings"
```

See also

- [Environment settings](#)
- [Synchronize your settings](#)
- [Personalize the Visual Studio IDE](#)
- [Visual Studio commands](#)

List Call Stack Command

10/21/2019 • 2 minutes to read • [Edit Online](#)

Displays the current call stack.

Syntax

```
Debug.ListCallStack [/Count:number] [/ShowTypes:yes|no]
[/ShowNames:yes|no] [/ShowValues:yes|no] [/ShowModule:yes|no]
[/ShowLineOffset:yes|no] [/ShowByteOffset:yes|no]
[/ShowLanguage:yes|no] [/IncludeCallsAcrossThreads:yes|no]
[/ShowExternalCode:yes|no] [Thread:n] [index]
```

Arguments

`index`

Optional. Sets the current stack frame and displays no output.

Switches

Each switch can be invoked using either its complete form or a short form.

`/Count: number` [or] `/C: number`

Optional. Maximum number of call stacks to display. The default value is unlimited.

`/ShowTypes: yes | no` [or] `/T: yes | no`

Optional. Specifies whether to display parameter types. Default value is `yes`.

`/ShowNames: yes | no` [or] `/N: yes | no`

Optional. Specifies whether to display parameter names. Default value is `yes`.

`/ShowValues: yes | no` [or] `/V: yes | no`

Optional. Specifies whether to display parameter values. Default value is `yes`.

`/ShowModule: yes | no` [or] `/M: yes | no`

Optional. Specifies whether to display the module name. Default value is `yes`.

`/ShowLineOffset: yes | no` [or] `/#: yes | no`

Optional. Specifies whether to display the line offset. Default value is `no`.

`/ShowByteOffset: yes | no` [or] `/B: yes | no`

Optional. Specifies whether to display the byte offset. Default value is `no`.

`/ShowLanguage: yes | no` [or] `/L: yes | no`

Optional. Specifies whether to display the language. Default value is `no`.

`/IncludeCallsAcrossThreads: yes | no` [or] `/I: yes | no`

Optional. Specifies whether to include calls to or from other threads. Default value is `no`.

/ShowExternalCode: `yes` | `no`

Optional. Specifies whether to display Just My Code for the callstack. When Just My Code is off, all non-user code is displayed. When Just My Code is on, non-user code is displayed as `[external]` in the callstack output.

Thread: `n`

Optional. Displays the callstack for thread `n`. If no thread is specified, displays the callstack for the current thread.

Remarks

Changes made to the arguments or switches apply to future invocations of this command. If you issue `Debug.ListCallStack` by itself, the entire call stack displays. If you specify an index, for example,

```
Debug.ListCallStack 2
```

then the current stack frame is set to that frame (in this case, the second frame).

You can also write this command using its pre-defined alias, `kb`. For example, you can enter

```
kb 2
```

to set the current stack frame to the second frame.

Example

```
>Debug.CallStack /Count:4 /ShowTypes:yes
```

See also

- [List Disassembly Command](#)
- [List Threads Command](#)
- [Visual Studio Commands](#)
- [Command Window](#)
- [Find/Command Box](#)
- [Visual Studio Command Aliases](#)

List Disassembly Command

10/21/2019 • 2 minutes to read • [Edit Online](#)

Begins the debug process and allows you to specify how errors are handled.

Syntax

```
Debug.ListDisassembly [/count:number] [/endaddress:expression]
[/codebytes:yes|no] [/source:yes|no] [/symbolnames:yes|no]
[/linenumbers:yes|no]
```

Switches

Each switch can be invoked using either its complete form or a short form.

/count: `number` [or] /c: `number` [or] /length: `number` [or] /l: `number`

Optional. Number of instructions to display. Default value is 8.

/endaddress: `expression` [or] /e: `expression`

Optional. Address at which to stop disassembly.

/codebytes: `yes` | `no` [or] /bytes: `yes` | `no` [or] /b: `yes` | `no`

Optional. Indicates whether to display code bytes. Default value is `no`.

/source: `yes` | `no` [or] /s: `yes` | `no`

Optional. Indicates whether to display source code. Default value is `no`.

/symbolnames: `yes` | `no` [or] /names: `yes` | `no` [or] /n: `yes` | `no`

Optional. Indicates whether to display symbols names. Default value is `yes`.

/linenumbers: `yes` | `no`]

Optional. Enables the viewing of line numbers associated with the source code. The /source switch must have a value of `yes` to use the /linenumbers switch.

Example

```
>Debug.ListDisassembly
```

See also

- [List Call Stack Command](#)
- [List Threads Command](#)
- [Visual Studio Commands](#)
- [Command Window](#)
- [Find/Command Box](#)

- Visual Studio Command Aliases

List Memory Command

10/21/2019 • 2 minutes to read • [Edit Online](#)

Displays the contents of the specified range of memory.

Syntax

```
Debug.ListMemory [/ANSI|Unicode] [/Count:number] [/Format:formattype]
[/Hex|Signed|Unsigned] [expression]
```

Arguments

`expression`

Optional. The memory address from which to begin displaying memory.

Switches

`/ANSI|Unicode`

Optional. Display the memory as characters corresponding to the bytes of memory, either ANSI or Unicode.

`/Count: number`

Optional. Determines how many bytes of memory to display, starting at `expression`.

`/Format: formattype`

Optional. Format type for viewing memory information in the **Memory** window; may be OneByte, TwoBytes, FourBytes, EightBytes, Float (32-bit), or Double (64-bit). If OneByte is used, `/Unicode` is unavailable.

`/Hex|Signed|Unsigned`

Optional. Specifies the format for viewing numbers: as signed, unsigned, or hexadecimal.

Remarks

Instead of writing out a complete **Debug.ListMemory** command with all switches, you can invoke the command using predefined aliases with certain switches preset to specified values. For example, instead of entering:

```
>Debug.ListMemory /Format:float /Count:30 /Unicode
```

you can write:

```
>df /Count:30 /Unicode
```

Here is a list of the available aliases for the **Debug.ListMemory** command:

ALIAS	COMMAND AND SWITCHES
d	Debug.ListMemory
da	Debug.ListMemory /Ansi
db	Debug.ListMemory /Format:OneByte
dc	Debug.ListMemory /Format:FourBytes /Ansi
dd	Debug.ListMemory /Format:FourBytes
df	Debug.ListMemory /Format:Float
dq	Debug.ListMemory /Format:EightBytes
du	Debug.ListMemory /Unicode

Example

```
>Debug.ListMemory /Format:float /Count:30 /Unicode
```

See also

- [List Call Stack Command](#)
- [List Threads Command](#)
- [Visual Studio Commands](#)
- [Command Window](#)
- [Find/Command Box](#)
- [Visual Studio Command Aliases](#)

List Modules Command

10/21/2019 • 2 minutes to read • [Edit Online](#)

Lists the modules for the current process.

Syntax

```
Debug.ListModules [/Address:yes|no] [/Name:yes|no] [/Order:yes|no]
[/Path:yes|no] [/Process:yes|no] [/SymbolFile:yes|no]
[/SymbolStatus:yes|no] [/Timestamp:yes|no] [/Version:yes|no]
```

Parameters

/Address: yes | no

Optional. Specifies whether to show the memory addresses of the modules. Default value is yes .

/Name: yes | no

Optional. Specifies whether to show the names of the modules. Default value is yes .

/Order: yes | no

Optional. Specifies whether to show the order of the modules. Default value is no .

/Path: yes | no

Optional. Specifies whether to show the paths of the modules. Default value is yes .

/Process: yes | no

Optional. Specifies whether to show the processes of the modules. Default value is no .

/SymbolFile: yes | no

Optional. Specifies whether to show the symbol files of the modules. Default value is no .

/SymbolStatus: yes | no

Optional. Specifies whether to show the symbol statuses of the modules. Default value is yes .

/Timestamp: yes | no

Optional. Specifies whether to show the timestamps of the modules. Default value is no .

/Version: yes | no

Optional. Specifies whether to show the versions of the modules. Default value is no .

Example

This example lists the module names, addresses, and timestamps for the current process.

```
Debug.ListModules /Address:yes /Name:yes /Order:no /Path:no /Process:no /SymbolFile:no /SymbolStatus:no
/Timestamp:yes /Version:no
```

See also

- [Visual Studio Commands](#)
- [Command Window](#)
- [How to: Use the Modules Window](#)

List Registers Command

10/21/2019 • 2 minutes to read • [Edit Online](#)

Displays the value of the selected registers and lets you modify the list of registers to show.

Syntax

```
Debug.ListRegisters [/Display [{register|registerGroup}...]] [/List]
[/Watch [{register|registerGroup}...]]
[/Unwatch [{register|registerGroup}...]]
```

Switches

/Display [{register | registerGroup}...]

Displays the values of the specified `register` or `registerGroup`. If no `register` or `registerGroup` is specified, the default list of registers is displayed. If no switch is specified, the behavior is the same. For example:

```
Debug.ListRegisters /Display eax
```

is equivalent to

```
Debug.ListRegisters eax
```

/List

Displays all register groups in the list.

/Watch [{register | registerGroup}...]

Adds one or more `register` or `registerGroup` values to the list.

/Unwatch [{register | registerGroup}...]

Removes one or more `register` or `registerGroup` values from the list.

Remarks

The alias `r` can be used in place of `Debug.ListRegisters`.

Example

This example uses the `Debug.ListRegisters` alias `r` to display the values of the register group `Flags`.

```
r /Display Flags
```

See also

- [Visual Studio Commands](#)
- [Debugging Basics: Registers Window](#)
- [How to: Use the Registers Window](#)

List Source Command

10/21/2019 • 2 minutes to read • [Edit Online](#)

Displays the specified lines of source code.

Syntax

```
Debug.ListSource [/Count:number] [/Current] [/File:filename]  
[/Line:number] [/ShowLineNumbers:yes|no]
```

Switches

/Count: `number`

Optional. Specifies the number of lines to display.

/Current

Optional. Shows the current line.

/File: `filename`

Optional. Path of the file to show. If no filename is specified, the command shows the source code for the line of the current statement.

/Line: `number`

Optional. Shows a specific line number.

/ShowLineNumbers: `yes|no`

Optional. Specifies whether to display line numbers.

Example

This example lists the source code from line 4 of the file Form1.vb, with line numbers visible.

```
Debug.ListSource /File:"C:\Visual Studio Projects\Form1.vb" /Line:4 /ShowLineNumbers:yes
```

See also

- [Visual Studio Commands](#)
- [Command Window](#)

List Threads Command

10/21/2019 • 2 minutes to read • [Edit Online](#)

Displays a list of the threads in the current program.

Syntax

```
Debug.ListThreads [index]
```

Arguments

`index`

Optional. Selects a thread by its index to be the current thread.

Remarks

When specified, the `index` argument marks the indicated thread as the current thread. An asterisk (*) is displayed in the list next to the current thread.

Example

```
>Debug.ListThreads
```

See also

- [List Call Stack Command](#)
- [List Disassembly Command](#)
- [Visual Studio Commands](#)
- [Command Window](#)
- [Find/Command Box](#)
- [Visual Studio Command Aliases](#)

Log Command window output command

10/18/2019 • 2 minutes to read • [Edit Online](#)

Copies all input and output from the **Command** window into a file.

Syntax

```
Tools.LogCommandWindowOutput [filename] [/on]/[off] [/overwrite]
```

Arguments

`filename`

Optional. The name of the log file. By default, the file is created in the user's profile folder. If the file name already exists, the log is appended to the end of the existing file. If no file is specified, the last file specified is used. If no previous file exists, a default log file is created, called cmdline.log.

TIP

To change the location where the log file is saved, enter the full path of the file, surrounded by quotation marks if the path contains any spaces.

Switches

`/on`

Optional. Starts the log for the **Command** window in the specified file and appends the file with the new information.

`/off`

Optional. Stops the log for the **Command** window.

`/overwrite`

Optional. If the file specified in the `filename` argument matches an existing file, the file is overwritten.

Remarks

If no file is specified, the file cmdline.log is created by default. By default, the alias for this command is Log.

Examples

This example creates a new log file, cmdlog, and starts the command log.

```
>Tools.LogCommandWindowOutput cmdlog
```

This example stops logging commands.

```
>Tools.LogCommandWindowOutput /off
```

This example resumes the logging of commands in the previously used log file.

```
>Tools.LogCommandWindowOutput /on
```

See also

- [Visual Studio commands](#)
- [Command window](#)
- [Find/Command box](#)
- [Visual Studio command aliases](#)

New File Command

10/21/2019 • 2 minutes to read • [Edit Online](#)

Creates a new file and opens it. The file appears under the Miscellaneous Files folder.

Syntax

```
File.NewFile [filename] [/t:templatename] [/editor:editorname]
```

Arguments

`filename`

Optional. Name for the file. If no name is supplied, a default name is provided. If no template name is listed, a text file is created.

Switches

`/t: templatename`

Optional. Specifies the type of file to be created.

The `/t: templatename` argument syntax mirrors the information found in the New File Dialog Box. Enter the category name followed by a backslash (`\`) and the template name, and enclose the entire string in quotation marks.

For example, to create a new Visual C++ source file, you would enter the following for the `/t: templatename` argument.

```
/t:"Visual C++\C++ File (.cpp)"
```

The example above indicates that the C++ File template is located under the Visual C++ category in the **New File** dialog box.

`/e: editorname`

Optional. Name of the editor in which the file will be opened. If the argument is specified but no editor name is supplied, the **Open With** dialog box appears.

The `/e: editorname` argument syntax uses the editor names as they appear in the Open With Dialog Box, enclosed in quotation marks.

For example, to open a file in the source code editor, you would enter the following for the `/e: editorname` argument.

```
/e:"Source Code (text) Editor"
```

Example

This example creates a new web page "test1.htm" and opens it in the source code editor.

```
>File.NewFile test1 /t:"General\HTML Page" /e:"Source Code (text) Editor"
```

See also

- [Visual Studio Commands](#)
- [Command Window](#)
- [Immediate Window](#)
- [Find/Command Box](#)
- [Visual Studio Command Aliases](#)

Open file command

10/18/2019 • 2 minutes to read • [Edit Online](#)

Opens an existing file and allows you to specify an editor.

Syntax

```
File.OpenFile filename [/e:editorname]
```

Arguments

`filename`

Required. The full or partial path and file name of the file to open. Paths containing spaces must be enclosed in quotation marks.

Switches

`/e: editorname`

Optional. Name of the editor in which the file will be opened. If the argument is specified but no editor name is supplied, the **Open With** dialog box appears.

The `/e: editorname` argument syntax uses the editor names as they appear in the Open With Dialog Box, enclosed in quotation marks.

For example, to open a file in the source code editor, you would enter the following for the `/e editorname` argument.

```
/e:"Source Code (text) Editor"
```

Remarks

As you enter a path, auto completion tries to locate the correct path and file name.

Example

This example opens the style file "Test1.css" in the source code editor.

```
>File.OpenFile "C:\My Projects\project1\Test1.css" /e:"Source Code (text) Editor"
```

See also

- [Visual Studio commands](#)
- [Command window](#)
- [Immediate window](#)
- [Find/Command box](#)

- [Visual Studio command aliases](#)

Open project command

10/18/2019 • 2 minutes to read • [Edit Online](#)

Opens an existing project or solution.

Syntax

```
File.OpenProject filename
```

Arguments

`filename`

Required. The full path and file name of the project or solution to open.

NOTE

The syntax for the `filename` argument requires that paths that contain spaces use quotation marks.

Remarks

Auto-completion tries to locate the correct path and file name as you type.

This command is not available while debugging.

Example

The following example opens the Visual Basic project **Test1**:

```
>File.OpenProject "C:\My Projects\Test1\Test1.vbproj"
```

See also

- [Visual Studio commands](#)
- [Command window](#)
- [Find/Command box](#)
- [Visual Studio command aliases](#)

Print command

10/21/2019 • 2 minutes to read • [Edit Online](#)

Evaluates an expression or displays specified text.

Syntax

```
>Debug.Print text
```

Arguments

text

Required. The expression to evaluate or the text to display.

Remarks

You can use the question mark (?) as an alias for this command. So, for example, the command

```
>Debug.Print expA
```

can also be written as

```
? expA
```

Both versions of this command return the current value of the expression `expA`.

Example

```
>Debug.Print DateTime.Now.Day
```

See also

- [Evaluate Statement Command](#)
- [Visual Studio Commands](#)
- [Command Window](#)
- [Find/Command Box](#)
- [Visual Studio Command Aliases](#)

Quick Watch Command

10/21/2019 • 2 minutes to read • [Edit Online](#)

Displays the selected or specified text in the Expression field of the [QuickWatch](#) window. You can use this dialog box to calculate the current value of a variable or expression recognized by the debugger, or the contents of a register. In addition, you can change the value of any non-const variable or the contents of any register.

Syntax

```
Debug.QuickWatchq [text]
```

Arguments

`text`

Optional. The text to add to the **QuickWatch** dialog box.

Remarks

If `text` is omitted, the currently selected text or word at the cursor is added to the Watch window.

Example

```
>Debug.QuickWatch
```

See also

- [Set a Watch on Variables using the Watch and QuickWatch Windows in Visual Studio](#)
- [Visual Studio Commands](#)
- [Command Window](#)
- [Find/Command Box](#)
- [Visual Studio Command Aliases](#)

Replace Command

10/21/2019 • 2 minutes to read • [Edit Online](#)

Replaces text in files using a subset of the options available on the **Replace in Files** tab of the **Find and Replace** window.

Syntax

```
Edit.Replace findwhat replacewith [/all] [/case]
[/doc|/proc|/open|/sel] [/hidden] [/options] [/reset] [/up]
[/wild|/regex] [/word]
```

Arguments

`findwhat`

Required. The text to match.

`replacewith`

Required. The text to substitute for the matched text.

Switches

`/all` or `/a`

Optional. Replaces all occurrences of the search text with the replacement text.

`/case` or `/c`

Optional. Matches occur only if when the uppercase and lowercase characters exactly match those specified in the `findwhat` argument.

`/doc` or `/d`

Optional. Searches the current document only. Specify only one of the available search scopes, `/doc`, `/proc`, `/open`, or `/sel`.

`/hidden` or `/h`

Optional. Searches concealed and collapsed text, such as the metadata of a design-time control, a hidden region of an outlined document, or a collapsed class or method.

`/open` or `/o`

Optional. Searches all open documents as if they were one document. Specify only one of the available search scopes, `/doc`, `/proc`, `/open`, or `/sel`.

`/options` or `/t`

Optional. Displays a list of the current find option settings and does not perform a search.

`/proc` or `/p`

Optional. Searches the current procedure only. Specify only one of the available search scopes, `/doc`, `/proc`,

/open , or /sel .

/regex or /r

Optional. Uses pre-defined special characters in the `findwhat` argument as notations that represent patterns of text rather than the literal characters. For a complete list of regular expression characters, see [Regular Expressions](#).

/reset or /e

Optional. Returns the find options to their default settings and does not perform a search.

/sel or /s

Optional. Searches the current selection only. Specify only one of the available search scopes, /doc , /proc , /open , or /sel .

/up or /u

Optional. Searches from the current location in the file toward the top of the file. By default, searches begin at the current location in the file and advance toward the bottom of the file.

/wild or /l

Optional. Uses pre-defined special characters in the `findwhat` argument as notations to represent a character or sequence of characters.

/word or /w

Optional. Searches only for whole words.

Example

This example replaces `btnSend` with `btnSubmit` in all open documents.

```
>Edit.Replace btnSend btnSubmit /open
```

See also

- [Finding and Replacing Text](#)
- [Command Window](#)
- [Find/Command Box](#)
- [Visual Studio Commands](#)
- [Visual Studio Command Aliases](#)

Replace In Files Command

10/21/2019 • 2 minutes to read • [Edit Online](#)

Replaces text in files using a subset of the options available on the **Replace in Files** tab of the **Find and Replace** window.

Syntax

```
Edit.ReplaceInFiles findwhat replacewith [/all] [/case]
[/ext:extensions] [/keep] [/lookin:searchpath] [/options] [/regex]
[/reset] [/stop] [/sub] [/text2] [/wild] [/word]
```

Arguments

`findwhat`

Required. The text to match.

`replacewith`

Required. The text to substitute for the matched text.

Switches

/all or /a

Optional. Replaces all occurrences of the search text with the replacement text.

/case or /c

Optional. Matches occur only if when the uppercase and lowercase characters exactly match those specified in the `findwhat` argument.

/ext: `extensions`

Optional. Specifies the file extensions for the files to be searched.

/keep or /k

Optional. Specifies that all modified files are left open.

/lookin: `searchpath`

Optional. Directory to search. If the path contains spaces, enclose the entire path in quotation marks.

/options or /t

Optional. Displays a list of the current find option settings and does not perform a search.

/regex or /r

Optional. Uses pre-defined special characters in the `findwhat` argument as notations that represent patterns of text rather than the literal characters. For a complete list of regular expression characters, see [Regular Expressions](#).

/reset or /e

Optional. Returns the find options to their default settings and does not perform a search.

/stop

Optional. Halts the current search operation if one is in progress. Replace ignores all other arguments when `/stop` has been specified. For example, to stop the current replacement you would enter the following:

```
>Edit.ReplaceInFiles /stop
```

/sub or /s

Optional. Searches the subfolders within the directory specified in the `/lookin: searchpath` argument.

/text2 or /2

Optional. Displays the results of the replacement in the **Find Results 2** window.

/wild or /l

Optional. Uses pre-defined special characters in the `findwhat` argument as notations to represent a character or sequence of characters.

/word or /w

Optional. Searches for only whole words.

Example

This example searches for `btnCancel` and replaces it with `btnReset` in all .cls files located in the folder "my visual studio projects" and displays the replacement information in the **Find Results 2** window.

```
>Edit.ReplaceInFiles btnCancel btnReset /lookin:"c:/my visual studio projects" /ext:.cls /text2
```

See also

- [Finding and Replacing Text](#)
- [Replace in Files](#)
- [Command Window](#)
- [Find/Command Box](#)
- [Visual Studio Commands](#)
- [Visual Studio Command Aliases](#)

Set Current Process

10/21/2019 • 2 minutes to read • [Edit Online](#)

Sets the specified process as the active process in the debugger.

Syntax

```
Debug.SetCurrentProcess index
```

Arguments

index

Required. The index of the process.

Remarks

You can attach to multiple processes when you are debugging, but only one process is active in the debugger at any given time. You can use the `SetCurrentProcess` command to set the active process.

Example

```
>Debug.SetCurrentProcess 1
```

See also

- [Visual Studio Commands](#)
- [Command Window](#)
- [Visual Studio Command Aliases](#)

Set Current Stack Frame Command

10/21/2019 • 2 minutes to read • [Edit Online](#)

Allows you to set a particular stack frame.

Syntax

```
Debug.SetCurrentStackFrame index
```

Arguments

`index`

Required. Selects a stack frame by its index.

Example

```
>Debug.SetCurrentStackFrame 1
```

See also

- [Visual Studio Commands](#)
- [Command Window](#)
- [Find/Command Box](#)
- [Visual Studio Command Aliases](#)

Set Current Thread Command

10/21/2019 • 2 minutes to read • [Edit Online](#)

Sets the specified thread as the current thread.

Syntax

```
Debug.SetCurrentThread index
```

Arguments

index

Required. Selects a thread by its index.

Example

```
>Debug.SetCurrentThread 1
```

See also

- [Visual Studio Commands](#)
- [Command Window](#)
- [Find/Command Box](#)
- [Visual Studio Command Aliases](#)

Set Radix Command

10/21/2019 • 2 minutes to read • [Edit Online](#)

Sets or returns the numeric base used to display integer values.

Syntax

```
Debug.SetRadix [10 | 16 | hex | dec]
```

Arguments

`10` or `16` or `hex` or `dec`

Optional. Indicates decimal (10 or dec) or hexadecimal (16 or hex). If an argument is omitted, then the current radix value is returned.

Example

This example sets the environment to display integer values in hexadecimal format.

```
>Debug.SetRadix hex
```

See also

- [Visual Studio Commands](#)
- [Command Window](#)
- [Find/Command Box](#)
- [Visual Studio Command Aliases](#)

Shell Command

10/21/2019 • 2 minutes to read • [Edit Online](#)

Launches executable programs from within Visual Studio.

Syntax

```
Tools.Shell [/command] [/output] [/dir:folder] path [args]
```

Arguments

`path`

Required. The path and file name of the file to execute or the document to open. A full path is required if the specified file is not in one of the directories in the PATH environment variable.

`args`

Optional. Any arguments to pass to the invoked program.

Switches

`/commandwindow` [or] `/command` [or] `/c` [or] `/cmd`

Optional. Specifies that the output for the executable is displayed in the **Command** window.

`/dir:` `folder` [or] `/d:` `folder`

Optional. Specifies the working directory to be set when the program is run.

`/outputwindow` [or] `/output` [or] `/out` [or] `/o`

Optional. Specifies that the output for the executable is displayed in the **Output** window.

Remarks

The `/dir` /`o` /`c` switches must be specified immediately after `Tools.Shell`. Anything specified after the name of the executable is passed to it as command line arguments.

The predefined alias `Shell` can be used in place of `Tools.Shell`.

Caution

If the `path` argument supplies the directory path as well as the file name, you should enclose the entire pathname in literal quotes (""""), as in the following:

```
Tools.Shell """C:\Program Files\SomeFile.exe"""
```

Each set of three double quotes ("""") is interpreted by the `Shell` processor as a single double quote character. Thus, the preceding example actually passes the following path string to the `Shell` command:

```
"C:\Program Files\SomeFile.exe"
```

Caution

If you do not enclose the path string in literal quotes ("\"), Windows will use only the portion of the string up to the first space. For example, if the path string above were not quoted properly, Windows would look for a file named "Program" located in the C:\ root directory. If a C:\Program.exe executable file were actually available, even one installed by illicit tampering, Windows would attempt to execute that program in place of the desired "c:\Program Files\SomeFile.exe" program.

Example

The following command uses xcopy.exe to copy the file `MyText.txt` into the `Text` folder. The output from xcopy.exe is displayed in both the **Command Window** and the **Output** window.

```
>Tools.Shell /o /c xcopy.exe c:\MyText.txt c:\Text\MyText.txt
```

See also

- [Visual Studio Commands](#)
- [Command Window](#)
- [Output Window](#)
- [Find/Command Box](#)
- [Visual Studio Command Aliases](#)

ShowWebBrowser Command

10/21/2019 • 2 minutes to read • [Edit Online](#)

Displays the URL you specify in a web browser window either within the integrated development environment (IDE) or external to the IDE.

Syntax

```
View.ShowWebBrowser URL [/new][/ext]
```

Arguments

URL

Required. URL (Uniform Resource Locator) for the website.

Switches

/new

Optional. Specifies that the page appears in a new instance of the web browser.

/ext

Optional. Specifies that the page appears in the default web browser outside of the IDE.

Remarks

The alias for the **ShowWebBrowser** command is **navigate** or **nav**.

Example

The following example displays the Microsoft Docs home page in a web browser outside of the IDE. If an instance of the web browser is already open, it is used; otherwise a new instance is launched.

```
>View.ShowWebBrowser https://docs.microsoft.com /ext
```

See also

- [Visual Studio Commands](#)
- [Command Window](#)
- [Find/Command Box](#)
- [Visual Studio Command Aliases](#)

Start Command

10/21/2019 • 2 minutes to read • [Edit Online](#)

Begins debugging the startup project.

Syntax

```
Debug.Start [address]
```

Arguments

address

Optional. The address at which the program suspends execution, similar to a breakpoint in source code. This argument is only valid in debug mode.

Remarks

The **Start** command, when executed, performs a RunToCursor operation to the specified address.

Example

This example starts the debugger and ignores any exceptions that occur.

```
>Debug.Start
```

See also

- [Visual Studio Commands](#)
- [Command Window](#)
- [Find/Command Box](#)
- [Visual Studio Command Aliases](#)

Symbol Path Command

10/21/2019 • 2 minutes to read • [Edit Online](#)

Sets the list of directories for the debugger to search for symbols.

Syntax

```
Debug.SymbolPath pathname1;pathname2;... pathnameN
```

Arguments

`pathname`

Optional. A semi-colon delimited list of paths for the debugger to search for symbols.

Remarks

If no `pathname` is specified, the command lists the current symbol paths.

Example

This example adds two paths to the list of symbol directories.

```
Debug.SymbolPath C:\Symbol Path 1;C:\Symbol Path 2
```

Example

This example displays a semi-colon delimited list of current symbol paths.

```
Debug.SymbolPath
```

See also

- [Command Window](#)
- [Visual Studio Commands](#)

Toggle Breakpoint Command

10/21/2019 • 2 minutes to read • [Edit Online](#)

Turns the breakpoint either on or off, depending on its current state, at the current location in the file.

Syntax

```
Debug.ToggleBreakpoint [text]
```

Arguments

text

Optional. If text is specified, the line is marked as a named breakpoint. Otherwise, the line is marked as an unnamed breakpoint, which is similar to what happens when you press F9.

Example

The following example toggles the current breakpoint.

```
>Debug.ToggleBreakpoint
```

See also

- [Visual Studio Commands](#)
- [Command Window](#)
- [Find/Command Box](#)
- [Visual Studio Command Aliases](#)

Watch Command

10/21/2019 • 2 minutes to read • [Edit Online](#)

Creates and opens a specified instance of a **Watch** window. You can use a **Watch** window to calculate the values of variables, expressions, and registers, to edit these values, and to save the results.

Syntax

```
Debug.Watch[index]
```

Arguments

`index`

Required. The instance number of the watch window.

Remarks

The `index` must be an integer. Valid values are 1, 2, 3, or 4.

Example

```
>Debug.Watch1
```

See also

- [Autos and Locals Windows](#)
- [Set a Watch on Variables using the Watch and QuickWatch Windows in Visual Studio](#)
- [Visual Studio Commands](#)
- [Command Window](#)
- [Find/Command Box](#)
- [Visual Studio Command Aliases](#)

View call hierarchy

10/18/2019 • 3 minutes to read • [Edit Online](#)

By viewing the call hierarchy for your code, you can navigate all calls to, and sometimes from, a selected method, property, or constructor. This enables you to better understand how code flows, and to evaluate the effects of changes to code. You can examine several levels of code to view complex chains of method calls and additional entry points to the code. This enables you to explore all possible execution paths.

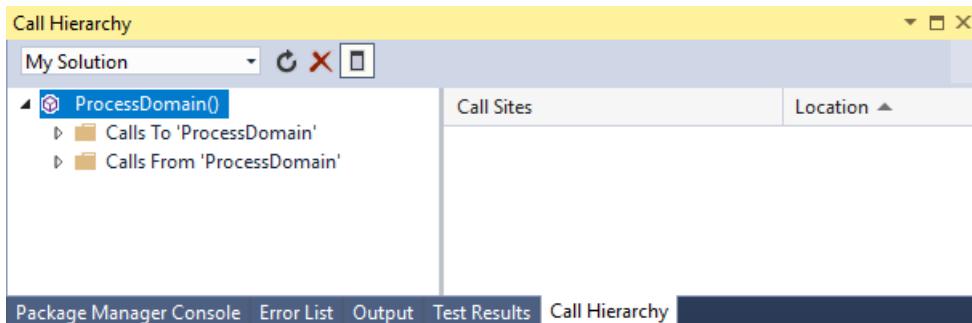
In Visual Studio, you can view a call hierarchy at design time. This means you don't have to set a breakpoint and start the debugger to view the run-time call stack.

Use the Call Hierarchy window

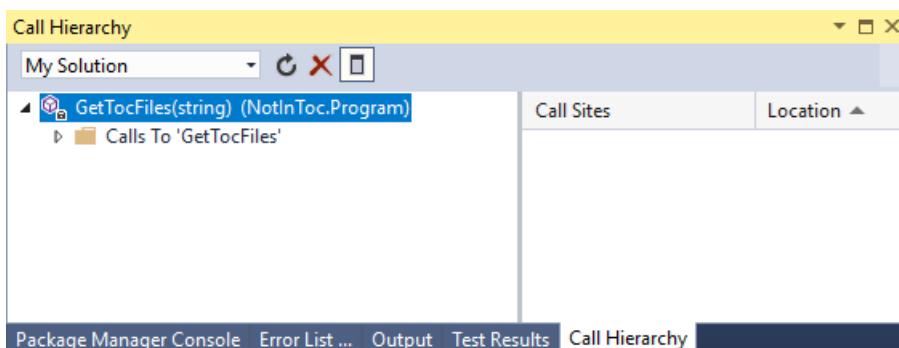
To display the **Call Hierarchy** window, right-click in the code editor on the name of a method, property, or constructor call, and then select **View Call Hierarchy**.

The member name appears in a tree view pane in the **Call Hierarchy** window. If you expand the member node, **Calls To member name**, and for C++, **Calls From member name**, subnodes appear.

For C++ code, you can see calls both to and from a member:

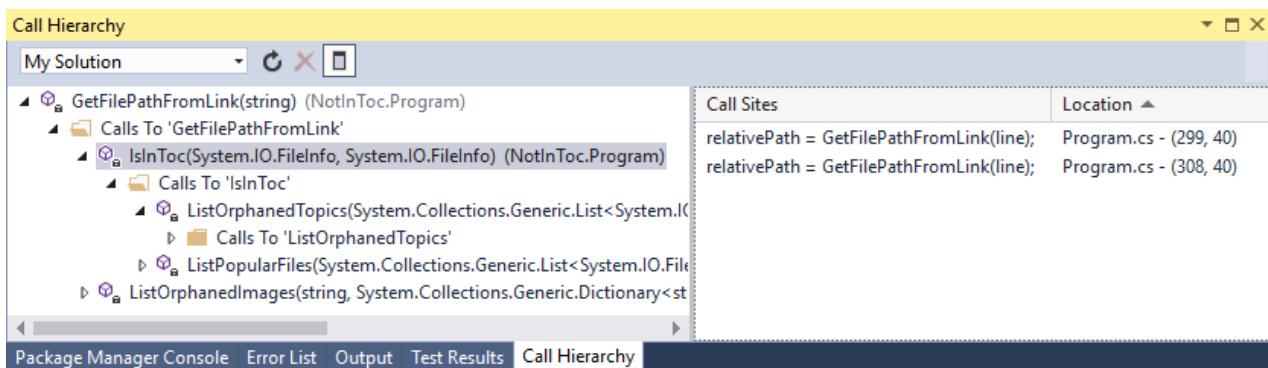


For C# and Visual Basic code, you can see calls to a member, but not calls from:



- If you expand the **Calls To** node, all members that call the selected member are displayed.
- For C++, if you expand the **Calls From** node, all members that are called by the selected member are displayed.

You can then expand each calling member to see its **Calls To**, and for C++, **Calls From** nodes. This enables you to navigate into the stack of callers, as shown in the following image:



For members that are defined as either virtual or abstract, an **Overrides method name** node appears. For interface members, an **Implements method name** node appears. These expandable nodes appear at the same level as the **Calls To** and **Calls From** nodes.

The **Search Scope** box on the toolbar contains choices for **My Solution**, **Current Project**, and **Current Document**.

When you select a child member in the **Call Hierarchy** tree view pane:

- The **Call Hierarchy** details pane displays all lines of code in which that child member is called from the parent member.
- The **Code Definition** window, if open, displays the code for the selected member (C++ only). For more information about this window, see [View the structure of code](#).

NOTE

The **Call Hierarchy** feature does not find method group references, which includes places where a method is added as an event handler or is assigned to a delegate. To find all references to a method, you can use the **Find All References** command.

Shortcut menu items

The following table describes several shortcut menu options that are available when you right-click a node in the tree view pane.

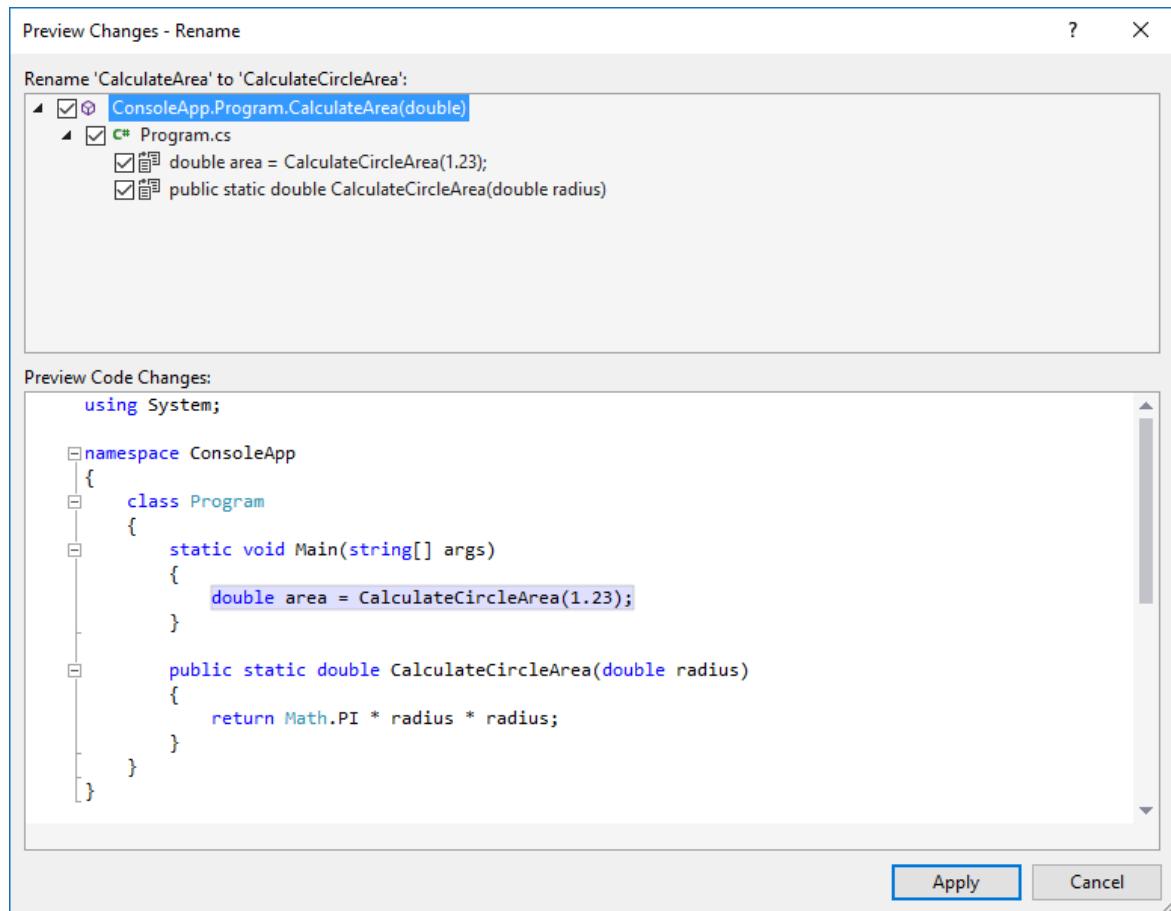
CONTEXT MENU ITEM	DESCRIPTION
Add As New Root	Adds the selected node to the tree view pane as a new root node. This enables you to focus your attention on a specific subtree.
Remove Root	Removes the selected root node from the tree view pane. This option is available only from a root node. You can also use the Remove Root toolbar button to remove the selected root node.
Go To Definition	Runs the Go To Definition command on the selected node. This navigates to the original definition for a member call or variable definition. To run the Go To Definition command, you can also double-click the selected node or press F12 on the selected node.

CONTEXT MENU ITEM	DESCRIPTION
Find All References	<p>Runs the Find All References command on the selected node. This finds all the lines of code in your project that reference a class or member.</p> <p>You can also use SHIFT+F12 to run the Find All References command on the selected node.</p>
Copy	Copies the contents of the selected node (but not its subnodes).
Refresh	Collapses the selected node so that re-expanding it displays current information.

Preview Changes window

10/18/2019 • 2 minutes to read • [Edit Online](#)

When using various *Quick Actions* or *Refactoring* tools in Visual Studio, it is often possible to preview changes that are going to be made to your project prior to accepting them. The **Preview Changes** window is where this is done. For example, here is the **Preview Changes** window showing what will be changed during a Rename refactor in a C# project:



The top half of the window shows the specific lines that will be changed, each with a checkbox. You can check or uncheck each checkbox if you want to selectively apply the refactoring to only specific lines.

The bottom half of the window shows the formatted code from the project that will be changed, with the affected areas highlighted. Selecting the specific line in the top half of the window will highlight the corresponding line in the bottom half. This allows you to quickly skip to the appropriate line and see the surrounding code.

After reviewing the changes, click the **Apply** button to commit those changes, or click the **Cancel** button to leave things as they were.

See also

- [Refactoring in Visual Studio](#)
- [Quick Actions](#)

Choose Toolbox items, WPF components

10/18/2019 • 2 minutes to read • [Edit Online](#)

This tab of the **Choose Toolbox Items** dialog box displays a list of Windows Presentation Foundation (WPF) controls available on your local computer. To display this list, select **Choose Toolbox Items** from the **Tools** menu to display the **Choose Toolbox Items** dialog box, and then select its **WPF Components** tab. To sort the components listed, select any column heading.

- When the check box next to a component is selected, an icon for that component will be displayed in the **Toolbox**.

TIP

To add a WPF control to a project document that's open for editing, drag its **Toolbox** icon onto the Design view surface. Default markup and code for the component are inserted into your project, ready for you to modify. For more information, see [Toolbox](#).

- When the check box next to a component is cleared, the corresponding icon will be removed from **Toolbox**.

NOTE

The .NET components installed on your computer remain available whether or not icons for them are displayed in the **Toolbox**.

The columns on the **WPF Components** tab contain the following information:

Name

Lists the names of WPF controls for which entries exist in your computer's registry.

Namespace

Displays the hierarchy of the [.NET API](#) namespace that defines the structure of the component. Sort on this column to list the available components within each .NET namespace installed on your computer.

Assembly Name

Displays the name of the .NET assembly that includes the namespace for each component. Sort on this column to list the namespaces contained in each .NET assembly installed on your computer.

Directory

Displays the location of the .NET assembly. The default location for all assemblies is the Global Assembly Cache. For further information on the Global Assembly Cache, see [Work with assemblies and the Global Assembly Cache](#).

UIElement List

Filter

Filters the list of WPF controls based on the string you provide in the text box. All matches from any of the four columns are shown.

Clear

Clears the filter string.

Browse

Opens the **Open** dialog box, which lets you navigate to assemblies which contain WPF controls. Use this to load assemblies which are not located in the Global Assembly Cache.

Language

Shows the localized language of the assembly which contains the selected WPF control.

Limitations

Adding a custom control or [UserControl](#) to the Toolbox has the following limitations:

- Works only for custom controls defined outside the current project.
- Does not update correctly when you change the solution configuration from Debug to Release, or from Release to Debug. This is because the reference is not a project reference, but is for the assembly on disk instead. If the control is part of the current solution, when you change from Debug to Release, your project continues to reference the Debug version of the control.

In addition, if design-time metadata is applied to the custom control and this metadata specifies that the [Microsoft.Windows.Design.ToolboxBrowsableAttribute](#) is set to `false`, the control does not appear in the Toolbox.

You can reference your controls directly in XAML view by mapping the namespace and assembly for your control.

See also

- [Toolbox](#)
- [Get started with WPF](#)

Code Snippet Picker

10/18/2019 • 2 minutes to read • [Edit Online](#)

The Visual Studio Code Editor provides a **Code Snippet Picker** that allows you, in a few mouse clicks, to insert ready-made blocks of code into the active document.

The procedure to display the **Code Snippet Picker** varies according to the language you are using.

- Visual Basic - Right-click at the desired location in the Code Editor to display the Shortcut menu, and select **Insert Snippet**.
- C# - Right-click at the desired location in the Code Editor to display the Shortcut menu, and click **Insert Snippet** or **Surround With**.
- C++ - The **Code Snippet Picker** is not available.
- F# - The **Code Snippet Picker** is not available.
- JavaScript - Right-click at the desired location in the Code Editor to display the Shortcut menu, and click **Insert Snippet** or **Surround With**.
- XML - Right-click at the desired location in the Code Editor to display the Shortcut menu, and click **Insert Snippet** or **Surround With**.
- HTML - Right-click at the desired location in the Code Editor to display the Shortcut menu, and click **Insert Snippet** or **Surround With**.
- SQL - Right-click at the desired location in the Code Editor to display the Shortcut menu, and click **Insert Snippet**.

In most Visual Studio development languages, you can use the **Code Snippets Manager** to add folders to the folder list that the **Code Snippet Picker** scans for XML snippet files. You can also create your own snippets to add to the list. For more information, see [Walkthrough: Creating a code snippet](#).

UIElement List

Item Name

An editable text field that displays the name of the item selected in the **Item List**. To perform an incremental search for the item you want, begin typing its name in this field. Continue adding letters until the desired item is selected in the **Item List**.

Item List

A list of code snippets available for insertion, or a list of folders containing code snippets. To insert a snippet or expand a folder, select the item you want and press Enter.

See also

- [Best Practices for Using Code Snippets](#)
- [Visual Basic IntelliSense Code Snippets](#)
- [Setting Bookmarks in Code](#)
- [How to: Use Surround-with Code Snippets](#)

Command Window

10/21/2019 • 3 minutes to read • [Edit Online](#)

The **Command** window is used to execute commands or aliases directly in the Visual Studio integrated development environment (IDE). You can execute both menu commands and commands that do not appear on any menu. To display the **Command** window, choose **Other Windows** from the **View** menu, and select **Command Window**.

Displaying the Values of Variables

To check the value of a variable `varA`, use the [Print Command](#):

```
>Debug.Print varA
```

The question mark (?) is an alias for `Debug.Print`, so this command can also be written:

```
>? varA
```

Both versions of this command will return the value of the variable `varA`.

Entering Commands

The greater than symbol (>) appears at the left edge of the Command window as a prompt for new lines. Use the UP ARROW and DOWN ARROW keys to scroll through previously issued commands.

TASK	SOLUTION	EXAMPLE
Evaluate an expression.	Preface the expression with a question mark (?).	? myvar
Switch to an Immediate window.	Enter <code>immed</code> into the window without the greater than sign (>)	immed
Switch back to the Command window from an Immediate window.	Enter <code>cmd</code> into the window.	>cmd

The following shortcuts help you navigate while in Command mode.

ACTION	CURSOR LOCATION	KEYBINDING
Cycle through the list of previously entered commands.	Input line	UP ARROW & DOWN ARROW
Scroll up the window.	Command window contents	CTRL+UP ARROW
Scroll down the window.	Command window contents	DOWN ARROW or CTRL+DOWN ARROW

TIP

You can copy all or part of a previous command to the input line by scrolling to it, highlighting all or part of it, and then pressing ENTER.

Mark Mode

When you click on any previous line in the **Command** window, you shift automatically into Mark mode. This allows you to select, edit, and copy the text of previous commands as you would in any text editor, and paste them into the current line.

The Equals (=) Sign

The window used to enter the `EvaluateStatement` command determines whether an equals sign (=) is interpreted as a comparison operator or as an assignment operator.

In the **Command** window, an equals sign (=) is interpreted as a comparison operator. You cannot use assignment operators in the **Command** window. So, for example, if the values of variables `varA` and `varB` are different, then the command `>Debug.EvaluateStatement(varA=varB)` will return a value of `False`.

In the **Immediate** window, by contrast, an equals sign (=) is interpreted as an assignment operator. So, for example, the command `>Debug.EvaluateStatement(varA=varB)` will assign to variable `varA` the value of variable `varB`.

Parameters, Switches, and Values

Some Visual Studio commands have required and optional arguments, switches and values. Certain rules apply when dealing with such commands. The following is an example of a rich command to clarify the terminology.

```
Edit.ReplaceInFiles /case /pattern:regex var[1-3]+ oldpar
```

In this example,

- `Edit.ReplaceInFiles` is the command
- `/case` and `/pattern:regex` are switches (prefaced with the slash [/] character)
- `regex` is the value of the `/pattern` switch; the `/case` switch has no value
- `var[1-3]+` and `oldpar` are parameters

NOTE

Any command, parameter, switch, or value that contains spaces must have double quotation marks on either side.

The position of switches and parameters can be interchanged freely on the command line with the exception of the **Shell** command, which requires its switches and parameters in a specific order.

Nearly every switch supported by a command has two forms: a short (one character) form and a long form. Multiple short-form switches can be combined into a group. For example, `/p /g /m` can be expressed alternately as `/pgm`.

If short-form switches are combined into a group and given a value, that value applies to every switch. For example, `/pgm:123` equates to `/p:123 /g:123 /m:123`. An error occurs if any of the switches in the group does not accept a value.

Escape Characters

A caret (^) character in a command line means that the character immediately following it is interpreted literally, rather than as a control character. This can be used to embed straight quotation marks ("), spaces, leading slashes, carets, or any other literal characters in a parameter or switch value, with the exception of switch names. For example,

```
>Edit.Find ^^t /regex
```

A caret functions the same whether it is inside or outside quotation marks. If a caret is the last character on the line, it is ignored. The example shown here demonstrates how to search for the pattern "`^t`".

Use Quotes for Path Names with Spaces

If, for example, you want to open a file that has a path containing spaces, you must put double quotes around the path or path segment that contains spaces: `C:\\"Program Files"` or `"C:\Program Files"`.

See also

- [Visual Studio Command Aliases](#)
- [Visual Studio Commands](#)

Convert dialog box

10/18/2019 • 2 minutes to read • [Edit Online](#)

The **Convert** dialog box was used in previous versions of Visual Studio and is now deprecated.

See also

- [Porting, Migrating, and Upgrading Visual Studio Projects](#)

Error List Window

10/18/2019 • 3 minutes to read • [Edit Online](#)

NOTE

The **Error List** displays information about a specific error message. You can copy the error number or error string text from the **Output** window. To display the **Output** window, press **Ctrl+Alt+O**. See [Output window](#).

The **Error List** window lets you perform the following tasks:

- Display the errors, warnings, and messages produced while you write code.
- Find syntax errors noted by IntelliSense.
- Find deployment errors, certain Static Analysis errors, and errors detected while applying Enterprise Template policies.
- Double-click any error message entry to open the file where the problem occurs, and move to the error location.
- Filter which entries are displayed, and which columns of information appear for each entry.
- Search for specific terms and scope the search to just the current project or document.

To display the **Error List**, choose **View > Error List**, or press **Ctrl+\+E**.

You can choose the **Errors**, **Warnings**, and **Messages** tabs to see different levels of information.

To sort the list, click any column header. To sort again by an additional column, hold down the **Shift** key and click another column header. To select which columns are displayed and which are hidden, choose **Show Columns** from the shortcut menu. To change the order in which columns are displayed, drag any column header to the left or right.

Error List Filters

There are two types of filter in two dropdown boxes, one on the right side of the toolbar and one to the left of the toolbar. The dropdown list on the left side of the toolbar specifies the set of code files to use (**Entire Solution**, **Open Documents**, **Current Project**, **Current Document**).

You can restrict the scope of the search to analyze and act on groups of errors. For example, you might want to focus on core errors that are preventing a project from compiling. The scoping options include:

1. **Open Documents:** Show errors, warnings, and messages for the open documents.
2. **Current Project:** Show errors, warnings, and messages from the project of the currently selected document in the **Editor** or the selected project in **Solution Explorer**.

NOTE

The filtered list of errors, warnings, and messages will change if the project of the currently selected document is different from the project selected in **Solution Explorer**.

3. **Current Document:** Show errors, warnings, and messages for the currently selected document in the

Editor or Solution Explorer.

If a filter is currently applied to the search result, the name of the filter appears in the **Error List** title bar. The **Errors**, **Warnings**, and **Messages** buttons then display the number of filtered items being shown along with the total number of items. For example, the buttons show "x of y Errors". If no filter is applied, the title bar says only "Error List".

The list on the right side of the toolbar specifies whether to show errors from the build (errors resulting from a build operation) or from IntelliSense (errors detected before running a build), or from both.

Search

Use the **Search Error List** text box on the right side of the **Error List** toolbar to find specific errors in the error list. You can search on any visible column in the error list, and search results are always sorted based on the column that has sort priority instead of on the query or the filter applied. If you choose the **Esc** key while the focus is in the **Error List**, you can clear the search term and filtered search results. You can also click the **X** on the right side of the text box to clear it.

Save

You can copy the error list and save it to a file. Select the errors you want to copy and right-click the selection, then on the context menu select **Copy**. You can then paste the errors into a file. If you paste the errors to an Excel spreadsheet, the fields appear as different columns.

UI Element List

Severity

Displays the different types of **Error List** entry (**Error**, **Message**, **Warning**, **Warning (active)**, **Warning (inactive)**).

Code

Displays the error code.

Description

Displays the text of the entry.

Project

Displays the name of the current project.

File

Displays the file name.

Line

Displays the line where the problem occurs.

File Properties, JavaScript

8/9/2019 • 2 minutes to read • [Edit Online](#)

You can use file properties to indicate what actions the project system should perform on the files. For example, you can set file properties to indicate whether a file should be added to the package as a resource file.

You can select any file in Solution Explorer and then examine its properties in the Properties window. JavaScript files have four properties: **Copy to Output Directory**, **Package Action**, **File Name**, and **File Path**.

File Properties

This section describes properties common to JavaScript files.

Copy to Output Directory Property

This property specifies the conditions under which the selected source file will be copied to the output directory. Select **Do not copy** if the file is never to be copied to the output directory. Select **Copy always** if the file is always to be copied to the output directory. Select **Copy if newer** if the file is to be copied only when it is newer than an existing file of the same name in the output directory.

Package Action

The **Package Action** property indicates what Visual Studio does with a file when a build is executed. **Package Action** can have one of several values:

- **None** - The file is not included in the package manifest. An example is a text file that contains documentation, such as a Readme file.
- **Content** - The file is included in the package manifest. For example, this setting is the default value for an .htm, .js, .css, image, audio, or video file.
- **Manifest** - The file is not included in the package manifest. Instead, the file is used for input when generating the package manifest. This is the default value for the package.appxmanifest file.
- **Resource** - The file is not included in the package manifest. Instead, the contents of the file are indexed in the Package Resource Index (PRI) that goes into the package manifest. It is typically used for resource files.

The default value for **Package Action** depends on the extension of the file that you add to the solution.

File Name Property

Displays the file name as a read-only value. To rename the file, you must right-click in Solution Explorer and select **Rename**.

Full Path Property

Displays the full path to the file as a read-only value. To change the path of the file, you can drag-and-drop the file in Solution Explorer.

Reference File Properties

This section describes properties common to files referenced from a UWP app built using JavaScript. When you select a reference such as a .winmd file, an SDK reference, a project-to-project reference, or an assembly reference in Solution Explorer, other properties may display in the Properties window, according to the file type.

Culture

Displays the language associated with the reference.

File Type

Displays the file type of the reference.

File Version

Displays the file version of the reference.

Identity

Displays the identity of the reference that is used in the project, which is stored in the project file.

Package

Displays the name of the package manifest associated with the reference.

Resolved Path

Displays the path to the reference that is used in the project.

SDK Path

Displays the path to the referenced SDK file.

Uri

Displays the URI that must be included in the project's HTML or JavaScript files to include the file as a source file.

Version

Displays the version of the reference.

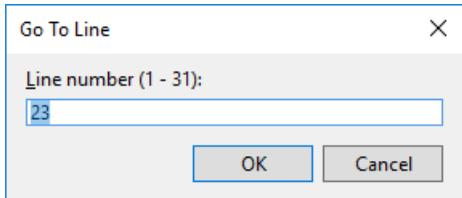
See also

- [Managing Project and Solution Properties](#)

Go To Line dialog box

10/21/2019 • 2 minutes to read • [Edit Online](#)

The **Go To Line** dialog box lets you move to a specific line in the active document. To access this dialog box, open a document for editing, and then select **Edit > Go To > Go To Line** or press **Ctrl+G**.



Line number (1 - <n>)

In the **Line number (1 - <n>)** box, enter the number of the line in the active document that you want to move to. The number entered must fall between 1 and the last line number in the current document.

See also

- [Find code using Go To commands](#)
- [Set bookmarks in code](#)
- [Find and replace text](#)
- [Features of the code editor](#)

Immediate window

10/18/2019 • 3 minutes to read • [Edit Online](#)

Use the **Immediate** window to debug and evaluate expressions, execute statements, and print variable values. The **Immediate** window evaluates expressions by building and using the currently selected project.

To display the **Immediate** window, open a project for editing, and then choose **Debug > Windows > Immediate** or press **Ctrl+Alt+I**. You can also enter **Debug.Immediate** in the **Command** window.

The **Immediate** window supports IntelliSense.

Display the values of variables

The **Immediate** window is particularly useful when you're debugging an app. For example, to check the value of a variable `varA`, you can use the [Print command](#):

```
>Debug.Print varA
```

The question mark (?) is an alias for `Debug.Print`, so this command can also be written:

```
? varA
```

Both versions of this command return the value of the variable `varA`.

TIP

To issue a Visual Studio command in the **Immediate** window, you must preface the command with a greater than sign (>). To enter multiple commands, switch to the [Command window](#).

Design-time expression evaluation

You can use the **Immediate** window to execute a function or subroutine at design time.

Execute a function at design time

1. Copy the following code into a Visual Basic console app:

```
Module Module1

    Sub Main()
        MyFunction(5)
    End Sub

    Function MyFunction(ByVal input as Integer) As Integer
        Return input * 2
    End Function

End Module
```

2. On the **Debug** menu, choose **Windows > Immediate**.

3. Type `?MyFunction(2)` in the **Immediate** window and press **Enter**.

The **Immediate** window runs `MyFunction` and displays `4`.

If the function or subroutine contains a breakpoint, Visual Studio breaks execution at the appropriate point. You can then use the debugger windows to examine your program state. For more information, see [Walkthrough: Debugging at Design Time](#).

You can't use design-time expression evaluation in project types that require starting up an execution environment, including Visual Studio Tools for Office projects, web projects, Smart Device projects, and SQL projects.

Design-time expression evaluation in multi-project solutions

When establishing the context for design-time expression evaluation, Visual Studio references the currently selected project in Solution Explorer. If no project is selected in Solution Explorer, Visual Studio attempts to evaluate the function against the startup project. If the function cannot be evaluated in the current context, you'll receive an error message. If you're attempting to evaluate a function in a project that's not the startup project for the solution and you receive an error, try selecting the project in Solution Explorer and attempt the evaluation again.

Enter commands

Enter the greater than sign (>) when issuing Visual Studio commands in the **Immediate** window. Use the **Up arrow** and **Down arrow** keys to scroll through your previously used commands.

TASK	SOLUTION	EXAMPLE
Evaluate an expression.	Preface the expression with a question mark (?).	<code>? a+b</code>
Temporarily enter Command mode while in Immediate mode (to execute a single command).	Enter the command, prefacing it with a greater than sign (>).	<code>>alias</code>
Switch to the Command window.	Enter <code>cmd</code> into the window, prefacing it with a greater than sign (>).	<code>>cmd</code>
Switch back to the Immediate window.	Enter <code>immed</code> into the window without the greater than sign (>).	<code>immed</code>

Mark mode

When you click on any previous line in the **Immediate** window, you shift automatically into Mark mode. This allows you to select, edit, and copy the text of previous commands as you would in any text editor, and paste them into the current line.

Examples

The following example shows four expressions and their result in the **Immediate** window for a Visual Basic project.

```
j = 2
Expression has been evaluated and has no value

? j
2

j = DateTime.Now.Day
Expression has been evaluated and has no value

? j
26
```

First-chance exception notifications

In some settings configurations, first-chance exception notifications are displayed in the **Immediate** window.

Toggle first-chance exception notifications in the Immediate window

1. On the **View** menu, click **Other Windows**, and click **Output**.
2. Right-click on the text area of the **Output** window, and then select or deselect **Exception Messages**.

See also

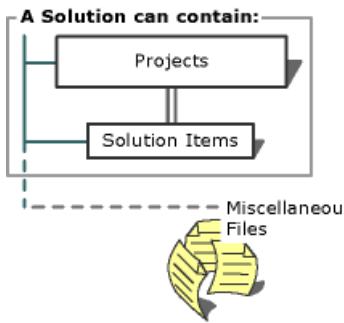
- [Navigating through Code with the Debugger](#)
- [Command Window](#)
- [First look at the debugger](#)
- [Walkthrough: Debugging at Design Time](#)
- [Visual Studio Command Aliases](#)
- [Using Regular Expressions in Visual Studio](#)

Miscellaneous files

10/18/2019 • 2 minutes to read • [Edit Online](#)

You might want to use the Visual Studio editor to work on files independently from a project or solution. While you have a solution open, you can open and modify files without adding them to a solution or to a project. Files you want to work with independently are called miscellaneous files. Miscellaneous files are external to solutions and projects, are not included in builds, and cannot be included with a solution under source control.

Opening files independently from a project or solution is useful for a variety of reasons. You might have a file that you want to view while developing a project-based solution but that's not integral to the solution's development. Common examples include development notes or instructions, database schema, and code clips. In addition, you might want to create a stand-alone file.



Solution Explorer can display a **Miscellaneous Files** folder for the files if the options for the folder are enabled. The options can be set from the [Documents, Environment, Options Dialog Box](#). After you close a miscellaneous file, it is not associated with any particular solution or project unless an option is enabled for that as well.

The **Miscellaneous Files** folder represents the files as links. Although this folder is not part of a solution, when you open a solution, some or all of the miscellaneous files that were opened when the solution was last closed are reopened, depending upon the settings for the folder.

NOTE

Some of the files that do not appear in the **Miscellaneous Files** folder are files that you cannot modify within the IDE, such as .zip files and .doc files. The IDE doesn't track files that can only be modified through an external editor.

Commands available in the IDE

The menus, toolbars, and the commands they contain change based on the format of the file you open. When you open a text file, for example, the Text Editor toolbar appears and its commands are available. If you then open an XML Schema file, the XML Schema toolbar appears. While editing your XML Schema, the Text Editor toolbar's commands (or the toolbar itself) are unavailable. The XML Schema is the active window and as such, has current selection context. When you switch between a project file and a miscellaneous file, all project-related commands disappear and only those that are directly related to the miscellaneous file appear.

Folder display options

You can set display options for the **Miscellaneous Files** folder so that the folder appears even though you have not opened any miscellaneous files. The solution file does not permanently manage a list of miscellaneous files. It uses an optional feature that allows it to remember a per-user, most recently used (MRU) list of files.

See also

- [Develop code in Visual Studio without projects or solutions](#)
- [Solutions and Projects](#)
- [Documents, Environment, Options Dialog Box](#)

Options dialog box (Visual Studio)

10/18/2019 • 2 minutes to read • [Edit Online](#)

The **Options** dialog box enables you to configure the integrated development environment (IDE) to your needs. For example, you can establish a default save location for your projects, alter the default appearance and behavior of windows, and create shortcuts for commonly used commands. There are also options specific to your development language and platform. You can access **Options** from the **Tools** menu.

Layout of the Options dialog box

The **Options** dialog box is divided into two parts: a navigation pane on the left and a display area on the right. The tree control in the navigation pane includes folder nodes, such as Environment, Text Editor, Projects and Solutions, and Source Control. Expand any folder node to list the pages of options that it contains. When you select the node for a particular page, its options appear in the display area.

Options for an IDE feature do not appear in the navigation pane until the feature is loaded into memory. Therefore, the same options might not be displayed as you begin a new session that were displayed as you ended the last. When you create a project or run a command that uses a particular application, nodes for relevant options are added to the Options dialog box. These added options will then remain available as long as the IDE feature remains in memory.

NOTE

Some settings collections scope the number of pages that appear in the navigation pane of the Options dialog box. You can choose to view all possible pages by selecting **Show all settings**.

How options are applied

Clicking **OK** in the **Options** dialog box saves all settings on all pages. Clicking **Cancel** on any page cancels all change requests, including any just made on other **Options** pages. Some changes to option settings, such as those made on [Fonts and Colors](#), [Environment](#), [Options Dialog Box](#), will only take effect after you close and reopen Visual Studio.

Show all settings

Selecting or unselecting **Show all settings** applies all changes you have made in the **Options** dialog box, even though you have not yet clicked **OK**.

See also

- [Customizing the Editor](#)

Options dialog box: Environment > General

10/18/2019 • 3 minutes to read • [Edit Online](#)

Use this page to change color themes, status bar settings, and file extension associations, among other options, for the integrated development environment (IDE). You can access the **Options** dialog box by opening the **Tools** menu, choosing **Options**, opening the **Environment** folder and then choosing the **General** page. If this page does not appear in the list, select the **Show all settings** check box in the **Options** dialog box.

Visual Experience

Color Theme

Choose the **Blue**, **Light**, **Dark**, or **Blue (Extra Contrast)** color theme for the IDE.

You can install additional predefined themes and create custom themes by downloading and installing the **Visual Studio Color Theme Editor** from the [Visual Studio Marketplace](#). After you install this tool, additional color themes appear in the **Color Theme** list box.

Apply title case styling to menu bar

Menus use title case styling by default. Uncheck this option to use all uppercase styling instead.

Optimize rendering for screens with different pixel densities (requires restart)

This option enables or disables per-monitor dots per inch (DPI) awareness (or PMA). When PMA is enabled, the Visual Studio user interface appears crisp in any monitor display scale factor and DPI configuration, including across multiple monitors. To enable PMA, you need Windows 10 April 2018 Update or later and .NET Framework 4.8 or later. (This option appears greyed out if those two prerequisites are not met.)

TIP

- Windows 10 has a setting that says **Let Windows try to fix apps so they're not blurry**. Turning that Windows setting **on** has negligible effect if you have the **Optimize rendering for screens with different pixel densities** option checked.
- Windows 10 also includes a **Program Compatibility Troubleshooter**. We don't recommend trying to fix the appearance of Visual Studio by using that troubleshooter.

Automatically adjust visual experience based on client performance

Specifies whether Visual Studio sets the adjustment to the visual experience automatically or you set the adjustment explicitly. This adjustment may change the display of colors from gradients to flat colors, or it may restrict the use of animations in menus or popup windows.

TIP

Windows 10 has a setting that says **Let Windows try to fix apps so they're not blurry**. Turning that setting **on** is recommended if Visual Studio appears blurry on your monitor. Consider upgrading to [Visual Studio 2019](#), which has significantly improved display clarity because it is a per-monitor dots per inch aware application.

Enable rich client experience

Enables the full visual experience of Visual Studio, including gradients and animations. Clear this option when

using Remote Desktop connections or older graphics adapters, because these features may have poor performance in those cases. This option is available only when you clear the **Automatically adjust visual experience based on client** option.

Use hardware graphics acceleration if available

Uses hardware graphics acceleration if it is available, rather than software acceleration.

Other

Items to show in Window menu

Customizes the number of windows that appear in the Windows list of the **Window** menu. Enter a number between 1 and 24. The default value is 10.

Items shown in recently used lists

Customizes the number of most recently used projects and files that appear on the **File** menu. Enter a number between 1 and 24. The default value is 10. This is an easy way to retrieve recently used projects and files.

Show status bar

Displays the status bar. The status bar is located at the bottom of the IDE window and displays information about the progress of ongoing operations.

Close button affects active tool window only

Specifies that when the **Close** button is clicked, only the tool window that has focus is closed and not all of the tool windows in the docked set. By default, this option is selected.

Auto Hide button affects active tool window only

Specifies that when the **Auto Hide** button is clicked, only the tool window that has focus is hidden automatically and not all of the tool windows in the docked set. By default, this option is not selected.

See also

- [Customize window layouts](#)

Accounts, Environment, Options dialog box

10/18/2019 • 2 minutes to read • [Edit Online](#)

Use this page to set various options related to the accounts you use to sign in to Visual Studio.

Personalization account

Synchronize settings across devices

Use this option to specify whether to synchronize your settings across multiple machines. For more information, see [Synchronized settings](#).

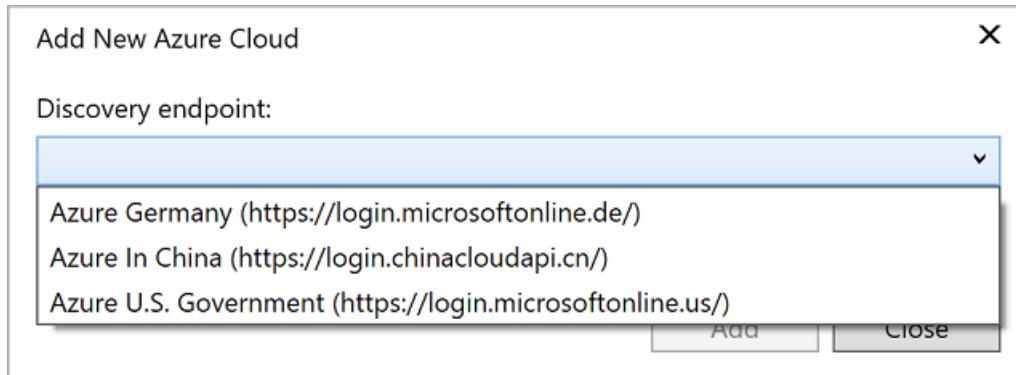
Enable device code flow

When this option is selected, the behavior of Visual Studio changes when you select **Add an account** on the **File > Account Settings** page. Instead of seeing the **Sign in to your account** page, you're presented with a dialog box that gives you a URL and a code to paste into a web browser to sign in. This option is useful in cases where you can't sign in to Visual Studio in the regular manner, for example, if you use an older version of Internet Explorer, or if your firewall restricts access. For more information, see [Work with multiple user accounts](#).

Registered Azure clouds

This section shows the Azure cloud instances that you have access to through one or more of the accounts you use to sign in to Visual Studio. For example, you might have access to a private instance of Azure in your company's data center. Or, you might have access to a sovereign or government instance of Azure such as Azure China 21 Vianet or Azure U.S. Government. The global Azure cloud instance appears by default in the list, and you cannot remove it.

Register an additional Azure cloud by choosing the **Add** button. The **Add New Azure Cloud** dialog lists several well-known Azure cloud instances you can connect to, or you can enter the URL to a private Azure endpoint.



After you register an additional Azure cloud, you can choose which Azure cloud you want to sign into when you sign into Visual Studio.

See also

- [Synchronize settings across multiple computers](#)
- [Sign in to Visual Studio](#)
- [Work with multiple user accounts](#)
- [Environment settings](#)

AutoRecover, Environment, Options dialog box

10/18/2019 • 2 minutes to read • [Edit Online](#)

Use this page in the **Options** dialog box to specify whether to automatically back up files or not. You can also specify if you want to restore modified files if Visual Studio shuts down unexpectedly.

Access this dialog box by selecting the **Tools** menu, selecting **Options**, and then selecting **Environment > AutoRecover**. If this page doesn't appear in the list, select **Show all settings** in the **Options** dialog box.

Save AutoRecover information every [n] minutes

Use this option to customize how often a file is automatically saved in the editor. For previously saved files, a copy of the file is saved in `%USERPROFILE%\Documents\Visual Studio\[version]\Backup Files\[projectname]`. If the file is new and you haven't saved it yet, the file is autosaved using a randomly generated file name.

Keep AutoRecover information for [n] days

Use this option to specify how long Visual Studio keeps files created for autorecovery.

See also

- [Options dialog box](#)

Options dialog box: Environment > Documents

10/18/2019 • 2 minutes to read • [Edit Online](#)

Use this page of the **Options** dialog box to control the display of documents in the integrated development environment (IDE) and manage external changes to documents and files. You can access this dialog box by clicking **Options** on the **Tools** menu and then selecting **Environment > Documents**.

Detect when file is changed outside the environment

When this option is selected, a message immediately notifies you of changes to an open file that have been made by an editor outside the IDE. The message lets you reload the file from storage.

Reload modified files unless there are unsaved changes

When you have **Detect when file is changed outside the environment** selected and an open file in the IDE changes outside the IDE, a warning message is generated by default. If this option is enabled, no warning appears and the document is reloaded in the IDE to pick up the external changes.

Allow editing of read-only files; warn when attempt to save

When this option is enabled, you can open and edit a read-only file. When you are finished, you must use the **Save As** command to save the file by a new name if you want to save a record of your changes.

Open file using directory of currently active document

When selected, this option specifies that the **Open File** dialog box displays the directory of the active document. When this option is cleared, the **Open File** dialog box displays the directory last used to open a file.

Check for consistent line endings on load

Select this option to have the editor scan the line endings in a file and display a message box if inconsistencies are detected in how line endings are formatted.

Display warning when global undo will modify edited files

Select this option to display a message box when the **Global Undo** command will roll back refactoring changes made in files that also were changed after the refactoring operation. Returning a file to its pre-refactoring state might discard subsequent changes made in the file.

Show Miscellaneous files in Solution Explorer

Select this option to display the **Miscellaneous Files** node in **Solution Explorer**. Miscellaneous files are files that are not associated with a project or solution but can appear in **Solution Explorer** for your convenience.

NOTE

Select this option to enable the **View in Browser** command on the **File** menu for web documents not included in the active web application.

Items saved in the Miscellaneous files project

Specifies the number of files to persist in the **Miscellaneous Files** folder of **Solution Explorer**. These files are listed even if they are no longer open in an editor. You can specify any whole number from 0 to 256. The default number is 0.

For example, if you set this option to 5 and have 10 miscellaneous files open, when you close all 10 files, the first 5 will still be shown in the **Miscellaneous Files** folder.

Save documents as Unicode when data cannot be saved in codepage

Select this option to cause files containing information incompatible with the selected codepage to be saved as Unicode by default.

See also

- [Miscellaneous Files](#)
- [Finding and Replacing Text](#)

Options dialog box: Environment > Extensions

10/18/2019 • 2 minutes to read • [Edit Online](#)

Use this page to set options for how Visual Studio performs updates and how extensions are updated and discovered.

Automatically check for updates

When checked, Visual Studio will check periodically for updates to itself, installed SDKs and tools, and extensions, and display notifications in the menu bar when updates are available. For more information, see [Update Visual Studio](#).

Automatically update extensions

When checked, updates to extensions are made without prompting. For more information, see [Finding and Using Visual Studio Extensions](#).

Load per user extensions when running as administrator

For more information, see [Finding and Using Visual Studio Extensions](#).

Additional Extension Galleries

An Enterprise feature that enables support for galleries of proprietary extensions. For more information, see [Private galleries](#).

Find and Replace, Environment, Options Dialog Box

10/18/2019 • 2 minutes to read • [Edit Online](#)

Use this page of the **Options** dialog box to control message boxes and other aspects of a find and replace operation. You can access this dialog box from the **Tools** menu by clicking **Options**, expanding **Environment**, and then clicking **Find and Replace**. If this page does not appear in the list, select **Show all setting** in the **Options** dialog box.

UIElement List

Display informational messages

Select this option to display all Find and Replace informational messages that have the **Always show this message** option. For example, if you chose not to display the message "Find reached the starting point of the search.", selecting this option would cause this informational message to appear again when you use Find and Replace.

If you do not want to see any informational messages for Find and Replace, clear this option.

When you have cleared the **Always show this message** option on some, but not all, **Find and Replace** informational messages, the **Display informational messages** check box appears to be filled but not selected. To restore all optional **Find and Replace** messages, clear this option and then select it again.

NOTE

This option does not affect any **Find and Replace** informational messages that do not display the **Always show this message** option.

Display warning messages

Select this option to display all cautionary Find and Replace messages that have the **Always show this message** option. For example, if you chose not to display the **Replace All** warning message that appears when you attempt to make replacements in files not currently opened for editing, selecting this option would cause this warning message to appear again when you attempt to Replace All.

If you do not want to see any cautionary messages for Find and Replace, clear this option.

When you have cleared the **Always show this message** option on some, but not all, **Find and Replace** warning messages, the **Display warning messages** check box appears to be filled but not selected. To restore all optional **Find and Replace** messages, clear this option and then select it again.

NOTE

This option does not affect any **Find and Replace** warning messages that do not display the **Always show this message** option.

Automatically populate Find What with text from the editor

Select this option to paste the text on either side of the current editor's insertion point into the **Find what** field when you select any view of the **Find and Replace** window from the **Edit** menu. Clear this option to use the last search pattern from the previous search as the **Find what** string.

See also

- [Finding and Replacing Text](#)

Fonts and Colors, Environment, Options Dialog Box

10/18/2019 • 16 minutes to read • [Edit Online](#)

The **Fonts and Colors** page of the **Options** dialog box lets you establish a custom font and color scheme for various user interface elements in the integrated development environment (IDE). You can access this dialog box by clicking **Tools > Options**, and then selecting **Environment > Fonts and Colors**.

Color scheme changes do not take effect during the session in which you make them. You can evaluate color changes by opening another instance of Visual Studio and producing the conditions under which you expect your changes to apply.

Show settings for

Lists all of the user interface elements for which you can change font and color schemes. After selecting an item from this list you can customize color settings for the item selected in **Display items**.

- **Text Editor**

Changes to font style, size, and color display settings for Text Editor affect the appearance of text in your default text editor. Documents opened in a text editor outside the IDE will not be affected by these settings.

- **Printer**

Changes to font style, size, and color display settings for Printer affect the appearance of text in printed documents.

NOTE

As needed, you can select a different default font for printing than that used for display in the text editor. This can be useful when printing code that contains both single-byte and double-byte characters.

- **Statement Completion**

Changes the font style and size for the text that appears in statement completion pop-up in the editor.

- **Editor Tooltip**

Changes the font style and size for the text that appears in ToolTips displayed in the editor.

- **Environment Font**

Changes the font style and size for all IDE user interface elements that do not already have a separate option in **Show settings for**.

For example, this option applies to the **Start Page** but does not affect the **Output** window.

- **[All Text Tool Windows]**

Changes to font style, size, and color display settings for this item affect the appearance of text in tool windows that have output panes in the IDE. For example, Output window, Command window, Immediate window, etc.

NOTE

Changes to the text of [All Text Tool Windows] items do not take effect during the session in which you make them. You can evaluate such changes by opening another instance of Visual Studio.

Use Defaults

Resets the font and color values of the list item selected in **Show settings for**. The **Use** button appears when other display schemes are available for selection. For example, you can choose from two schemes for the Printer.

Font (bold type indicates fixed-width fonts)

Lists all the fonts installed on your system. When the drop-down menu first appears, the current font for the element selected in the **Show settings for** field is highlighted. Fixed fonts — which are easier to align in the editor — appear in bold.

Size

Lists available point sizes for the highlighted font. Changing the size of the font affects all **Display items** for the **Show settings for** selection.

Display items

Lists the items for which you can modify the foreground and background color.

NOTE

Plain Text is the default display item. As such, properties assigned to **PlainText** will be overridden by properties assigned to other display items. For example, if you assign the color blue to **PlainText** and the color green to **Identifier**, all identifiers will appear in green. In this example, **Identifier** properties override **PlainText** properties.

Some of display items include:

DISPLAY ITEM	DESCRIPTION
Plain Text	Text in the editor.
Selected Text	Text that is included in the current selection when the editor has focus.
Inactive Selected Text	Text that is included in the current selection when the editor has lost focus.
Indicator Margin	The margin at the left of the Code Editor where breakpoints and bookmark icons are displayed.
Line Numbers	Optional numbers that appear next to each line of code
Visible White Space	Spaces, tabs and word wrap indicators
Bookmark	Lines that have bookmarks. Bookmark is only visible if the indicator margin is disabled.
Brace Matching (Highlight)	Highlighting that is typically bold formatting for matching braces.

DISPLAY ITEM	DESCRIPTION
Brace Matching (Rectangle)	Highlighting that is typically a grey rectangle in the background.
Breakpoint (Disabled)	Not used.
Breakpoint (Enabled)	Specifies the highlight color for statements or lines containing simple breakpoints. This option is applicable only if statement-level breakpoints are active or the Highlight entire source line for breakpoints or current statement option is selected on General, Debugging, Options Dialog Box .
Breakpoint (Error)	Specifies the highlight color for statements or lines containing breakpoints that are in an error state. Applicable only if statement-level breakpoints are active or the Highlight entire source line for breakpoints or current statement option is selected on General, Debugging, Options Dialog Box .
Breakpoint (Warning)	Specifies the highlight color for statements or lines containing breakpoints that are in a warning state. Applicable only if statement-level breakpoints are active or the Highlight entire source line for breakpoints or current statement option is selected on General, Debugging, Options Dialog Box .
Breakpoint - Advanced (Disabled)	Specifies the highlight color for statements or lines containing disabled conditional or hit-counted breakpoints. Applicable only if statement-level breakpoints are active or the Highlight entire source line for breakpoints or current statement option is selected on General, Debugging, Options Dialog Box .
Breakpoint - Advanced (Enabled)	Specifies the highlight color for statements or lines containing conditional or hit-counted breakpoints. Applicable only if statement-level breakpoints are active or the Highlight entire source line for breakpoints or current statement option is selected on General, Debugging, Options Dialog Box .
Breakpoint - Advanced (Error)	Specifies the highlight color for statements or lines containing conditional or hit-counted breakpoints that are in an error state. Applicable only if statement-level breakpoints are active or the Highlight entire source line for breakpoints or current statement option is selected on General, Debugging, Options Dialog Box .
Breakpoint - Advanced (Warning)	Specifies the highlight color for statements or lines containing conditional or hit-counted breakpoints that are in a warning state. Applicable only if statement-level breakpoints are active or the Highlight entire source line for breakpoints or current statement option is selected on General, Debugging, Options Dialog Box .
Breakpoint - Mapped (Disabled)	Specifies the highlight color for statements or lines containing disabled mapped breakpoints. Applicable for ASP or ASP.NET debugging if statement-level breakpoints are active or the Highlight entire source line for breakpoints or current statement option is selected on General, Debugging, Options Dialog Box .

DISPLAY ITEM	DESCRIPTION
Breakpoint - Mapped (Enabled)	Specifies the highlight color for statements or lines containing mapped breakpoints. Applicable for ASP or ASP.NET debugging if statement-level breakpoints are active or the Highlight entire source line for breakpoints or current statement option is selected on General, Debugging, Options Dialog Box .
Breakpoint - Mapped (Error)	Specifies the highlight color for statements or lines containing mapped breakpoints in an error state. Applicable for ASP or ASP.NET debugging if statement-level breakpoints are active or the Highlight entire source line for breakpoints or current statement option is selected on General, Debugging, Options Dialog Box .
Breakpoint - Mapped (Warning)	Specifies the highlight color for statements or lines containing mapped breakpoints in a warning state. Applicable for ASP or ASP.NET debugging if statement-level breakpoints are active or the Highlight entire source line for breakpoints or current statement option is selected on General, Debugging, Options Dialog Box .
C/C++ User Keywords	A constant within a particular code file defined by means of the <code>#define</code> directive.
Call Return	Specifies the highlight color for source statements or lines that indicate call return points when context is switched to a non-top stack frame when debugging.
Code Snippet Dependent Field	A field that will be updated when the current editable field is modified.
Code Snippet Field	Editable field when a code snippet is active.
Collapsible Text	A block of text or code that can be toggled in and out of view within the Code Editor.
Comment	Code comments.
Compiler Error	Blue squiggles in the editor indicating a compiler error.
Coverage Not Touched Area	Code that has not been covered by a unit test.
Coverage Partially Touched Area	Code that has been partially covered by a unit test.
Coverage Touched Area	Code that has been completely covered by a unit test.
CSS Comment	A comment in Cascading Style Sheets. For example: /* comment */
CSS Keyword	Keywords in the Cascading Style Sheet.
CSS Property Name	The name of a property, such as Background.

DISPLAY ITEM	DESCRIPTION
CSS Property Value	The value assigned to a property such as blue.
CSS Selector	A string that identifies what elements the corresponding rule applies to. A selector can either be a simple selector, such a 'H1', or a contextual selector, such as 'H1 B', which consists of several simple selectors.
CSS String Value	A string in Cascading Style Sheets.
Current list location	Current line navigated to in a list tool window, such as the Output window or Find Results windows.
Current Statement	Specifies the highlight color for the source statement or line that indicates the current step position when debugging.
Debugger Data Changed	The color of text used to display changed data inside the Registers and Memory windows.
Definition Window Background	The background color of the Code Definition window.
Definition Window Current Match	The current definition in the Code Definition window.
Disassembly File Name	The color of text used to display file name breaks inside the Disassembly window.
Disassembly Source	The color of text used to display source lines inside the Disassembly window.
Disassembly Symbol	The color of text used to display symbol names inside the Disassembly window.
Disassembly Text	The color of text used to display op-code and data inside the Disassembly window.
Excluded Code	Code that is not to be compiled, per a conditional preprocessor directive such as <code>#if</code> .
Identifier	Identifiers in code such as the class names, methods names, and variable names.
Keyword	Keywords for the given language that are reserved. For example: class and namespace.
Memory Address	The color of text used to display the address column inside the Memory window.
Memory Changed	The color of text used to display changed data inside the Memory window.
Memory Data	The color of text used to display data inside the Memory window.

DISPLAY ITEM	DESCRIPTION
Memory Unreadable	The color of text used to display unreadable memory areas within the Memory window.
Number	A number in code that represents an actual numeric value.
Operator	Operators such as +, -, and !=.
Other Error	Other error types not covered by other error squiggles. Currently, this includes rude edits in Edit and Continue.
Preprocessor Keyword	Keywords used by the preprocessor such as #include.
Read-Only Region	Code that cannot be edited. For example code displayed in the Code Definition View window or code that cannot be modified during Edit and Continue.
Refactoring Background	Background color of the Preview Changes dialog box.
Refactoring Current Field	Background color of the current element to be refactored in the Preview Changes dialog box.
Refactoring Dependent Field	Color of references of the element to be refactored in the Preview Changes dialog box.
Register Data	The color of text used to display data inside the Registers window.
Register NAT	The color of text used to display unrecognized data and objects inside the Registers window.
Smart Tag	Used to denote the outline when smart tags are invoked.
SQL DML Marker	Applies to the Transact-SQL editor. DML statements in this editor are marked with a bounding blue box by default.
Stale Code	Superseded code awaiting an update. In some cases, Edit and Continue cannot apply code changes immediately, but will apply them later as you continue debugging. This occurs if you edit a function that must call the function currently executing, or if you add more than 64 bytes of new variables to a function waiting on the call stack. When this happens, the debugger displays a "Stale Code Warning" dialog box, and the superseded code continues to execute until the function in question finishes and is called again. Edit and Continue applies the code changes at that time.
String	String literals.
String (C# @ Verbatim)	String literals in C# that are interpreted verbatim. For example: @ "x"
Syntax Error	Parse errors.

DISPLAY ITEM	DESCRIPTION
Task List Shortcut	If a Task List shortcut is added to a line, and the indicator margin is disabled, the line will be highlighted.
Tracepoint (Disabled)	Not used.
Tracepoint (Enabled)	Specifies the highlight color for statements or lines containing simple tracepoints. This option is applicable only if statement-level tracepoints are active or the Highlight entire source line for breakpoints or current statement option is selected on General, Debugging, Options Dialog Box .
Tracepoint (error)	Specifies the highlight color for statements or lines containing tracepoints that are in an error state. This option is applicable only if statement-level tracepoints are active or the Highlight entire source line for breakpoints or current statement option is selected on General, Debugging, Options Dialog Box .
Tracepoint (Warning)	Specifies the highlight color for statements or lines containing tracepoints that are in a warning state. This option is applicable only if statement-level tracepoints are active or the Highlight entire source line for breakpoints or current statement option is selected on General, Debugging, Options Dialog Box .
Tracepoint - Advanced (Disabled)	Specifies the highlight color for statements or lines containing disabled conditional or hit-counted tracepoints. This option is applicable only if statement-level tracepoints are active or the Highlight entire source line for breakpoints or current statement option is selected on General, Debugging, Options Dialog Box .
Tracepoint - Advanced (Enabled)	Specifies the highlight color for statements or lines containing conditional or hit-counted tracepoints. This option is applicable only if statement-level tracepoints are active or the Highlight entire source line for breakpoints or current statement option is selected on General, Debugging, Options Dialog Box .
Tracepoint - Advanced (Error)	Specifies the highlight color for statements or lines containing conditional or hit-counted tracepoints that are in an error state. This option is applicable only if statement-level tracepoints are active or the Highlight entire source line for breakpoints or current statement option is selected on General, Debugging, Options Dialog Box .
Tracepoint - Advanced (Warning)	Specifies the highlight color for statements or lines containing conditional or hit-counted tracepoints that are in a warning state. This option is applicable only if statement-level tracepoints are active or the Highlight entire source line for breakpoints or current statement option is selected on General, Debugging, Options Dialog Box .

DISPLAY ITEM	DESCRIPTION
Tracepoint - Mapped (Disabled)	Specifies the highlight color for statements or lines containing disabled mapped tracepoints. Applicable for ASP or ASP.NET debugging if statement-level breakpoints are active or the Highlight entire source line for breakpoints or current statement option is selected on General, Debugging, Options Dialog Box .
Tracepoint - Mapped (Enabled)	Specifies the highlight color for statements or lines containing mapped tracepoints. Applicable for ASP or ASP.NET debugging if statement-level breakpoints are active or the Highlight entire source line for breakpoints or current statement option is selected on General, Debugging, Options Dialog Box .
Tracepoint - Mapped (Error)	Specifies the highlight color for statements or lines containing mapped tracepoints in an error state. Applicable for ASP or ASP.NET debugging if statement-level breakpoints are active or the Highlight entire source line for breakpoints or current statement option is selected on General, Debugging, Options Dialog Box .
Tracepoint - Mapped (Warning)	Specifies the highlight color for statements or lines containing mapped tracepoints in a warning state. Applicable for ASP or ASP.NET debugging if statement-level breakpoints are active or the Highlight entire source line for breakpoints or current statement option is selected on General, Debugging, Options Dialog Box .
Track Changes after save	Lines of code that have been modified since the file was opened but are saved to disk.
Track Changes before save	Lines of code that have been modified since the file was opened but are not saved to disk.
User Types	Types defined by users.
User Types (Delegates)	Type color for delegates.
User Types (Enums)	Type color used for enums.
User Types (Interfaces)	Type color for interfaces.
User Types (Value types)	Type color for value types such as structs in C#.
Visual Basic Read Only Marker	A marker specific to Visual Basic used for designating EnC, such as exception regions, a method definition, and non-leaf call frames.
Warning	Compiler warnings.
Warning Lines Path	Used for Static Analysis warning lines.
XML Attribute	Attribute names.
XML Attribute Quotes	The quote characters for XML attributes.

DISPLAY ITEM	DESCRIPTION
XML Attribute Value	Contents of XML attributes.
XML Cdata Section	Contents of <![CDATA[...]]>.
XML Comment	The contents of <!-- -->.
XML Delimiter	XML Syntax delimiters, including <, <?, <!, <!--, -->, ?>, <![]>, and [,].
XML Doc Attribute	The value of an xml documentation attribute, such as <param name="I"> where the "I" is colorized.
XML Doc Comment	The comments enclosed in the xml documentation comments.
XML Doc Tag	The tags in XML doc comments, such as /// <summary>.
XML Keyword	DTD keywords such as CDATA, IDREF, and NDATA.
XML Name	Element names and Processing Instructions target name.
XML Processing Instruction	Contents of Processing Instructions, not including target name.
XML Text	Plain text element content.
XSLT Keyword	XSLT element names.

Item foreground

Lists the available colors you can choose for the foreground of the item selected in **Display items**. Because some items are related, and should therefore maintain a consistent display scheme, changing the foreground color of text also changes the defaults for elements such as Compiler Error, Keyword, or Operator.

Automatic

Items can inherit the foreground color from other display items such as **Plain Text**. Using this option, when you change the color of an inherited display item, the color of the related display items also change automatically. For example, if you selected the **Automatic** value for **Compiler Error** and later changed the color of **Plain Text** to Red, **Compiler Error** would also automatically inherit the color Red.

Default

The color that appears for the item the first time you open Visual Studio. Clicking the **Use Defaults** button resets to this color.

Custom

Displays the Color dialog box to allow you to set a custom color for the item selected in the Display items list.

NOTE

Your ability to define custom colors may be limited by the color settings for your computer display. For example, if your computer is set to display 256 colors and you select a custom color from the **Color** dialog box, the IDE defaults to the closest available **Basic color** and displays the color black in the **Color** preview box.

Item background

Provides a color palette from which you can choose a background color for the item selected in **Display items**. Because some items are related, and should therefore maintain a consistent display scheme, changing the background color of text also changes the defaults for elements such as Compiler Error, Keyword, or Operator.

Automatic

Items can inherit the background color from other display items such as **Plain Text**. Using this option, when you change the color of an inherited display item, the color of the related display items also change automatically. For example, if you selected the **Automatic** value for **Compiler Error** and later changed the color of **Plain Text** to Red, **Compiler Error** would also automatically inherit the color Red.

Default

The color that appears for the item the first time you open Visual Studio. Clicking the **Use Defaults** button resets to this color.

Custom

Displays the Color dialog box to allow you to set a custom color for the item selected in the Display items list.

Bold

Select this option to display the text of selected **Display items** in bold text. Bold text is easier to identify in the editor.

Sample

Displays a sample of the font style, size, and color scheme for the **Show settings for** and **Display items** selected. You can use this box to preview the results as you experiment with different formatting options.

See also

- [Options Dialog Box](#)
- [How to: Change Fonts and Colors](#)

Options dialog box: Environment > Import and Export Settings

10/18/2019 • 2 minutes to read • [Edit Online](#)

Use this page to specify where your user settings file is saved. For more information about settings, see [Personalize the Visual Studio IDE](#).

Options dialog box: Environment > International Settings

10/18/2019 • 2 minutes to read • [Edit Online](#)

The International Settings page lets you change the default language when you have more than one language version of the integrated development environment (IDE) installed on your machine. You can access this dialog box by selecting **Options** from the **Tools** menu and then choosing **International Settings** from the **Environment** folder.

Language

Lists the available languages for the installed product language versions. If multiple languages of products or a mixed language installation of products share the environment, the language selection is changed to **Same as Microsoft Windows**.

Caution

In a system with multiple languages installed, the Visual C++ build tools (cl.exe, link.exe, nmake.exe, bscmake.exe and related files) are not affected by this setting. These tools use the version for the last language installed. The build tools for the previously installed language are overwritten because the Visual C++ build tools do not use the satellite DLL model.

See also

- [Install language packs](#)

Options dialog box: Environment > Keyboard

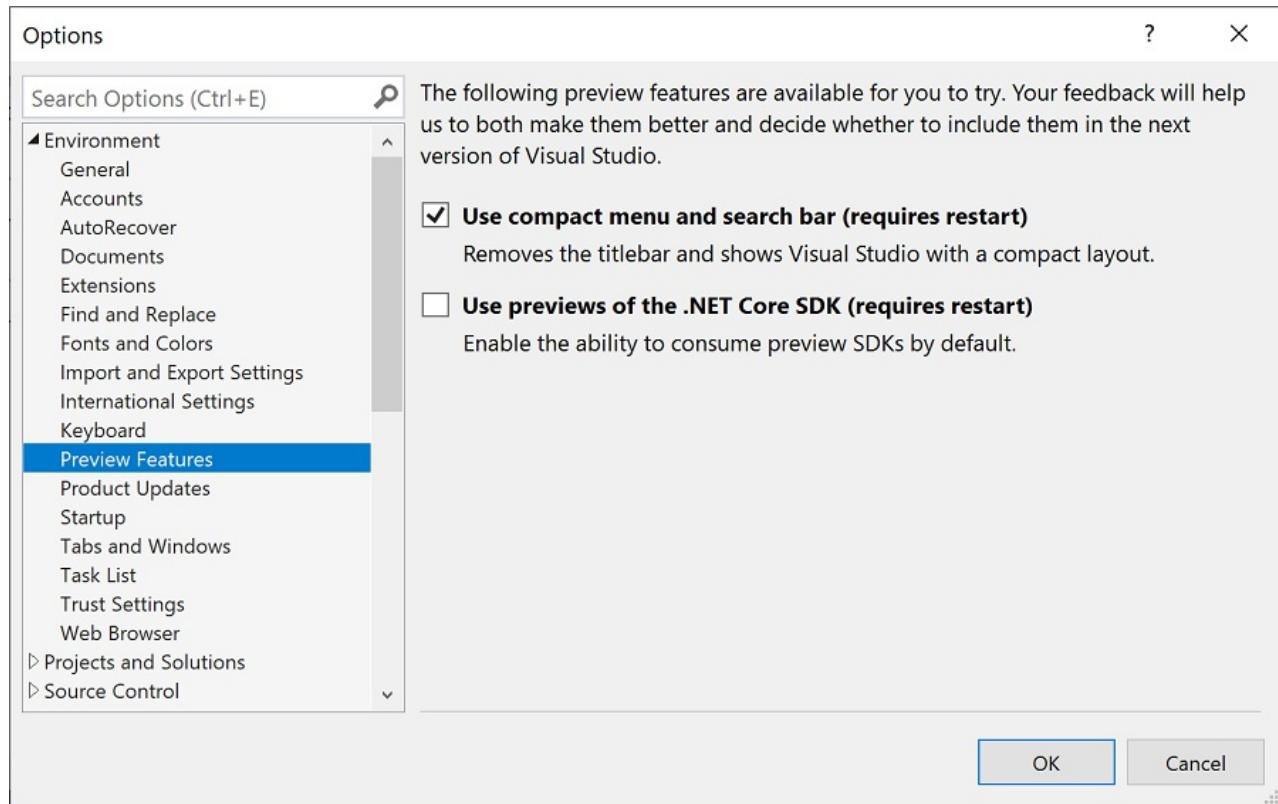
10/18/2019 • 2 minutes to read • [Edit Online](#)

Use this page to set keyboard mappings. For more information about mappings, see [Identify and customize keyboard shortcuts](#).

Options dialog box: Environment > Preview Features

10/18/2019 • 2 minutes to read • [Edit Online](#)

Use this page of the **Options** dialog box to enable or disable preview features in Visual Studio. The **Preview Features** options page changes frequently as different preview features are made available in Visual Studio.



You can access this dialog box by clicking **Options** on the **Tools** menu and then selecting the **Environment > Preview Features** page. Or, enter **preview features** into the Visual Studio **Search** box (press **Ctrl+Q** to move focus to the **Search** box).

Notifications, Environment, Options dialog box

10/18/2019 • 2 minutes to read • [Edit Online](#)

Use this option to stop ignoring any notifications that you previously chose to ignore through the **Notifications** dialog. For more information, see [Update Visual Studio](#).

Quick Launch, Environment, Options Dialog Box

10/18/2019 • 2 minutes to read • [Edit Online](#)

You can use **Quick Launch** to quickly search and execute actions for IDE assets such as options, templates, menus. You can't use **Quick Launch** to search for code and symbols. The **Quick Launch** search box is located at the top-right corner of the menu bar and is accessible by pressing **Ctrl+Q**. Type your search string in the box. To search for strings that contain @, use '@@'.

Quick Launch is enabled by default when you install Visual Studio. On the menu bar, you can show or hide **Quick Launch** by choosing **Tools > Options**. Expand the **Environments** node, and then choose **Quick Launch**. Select or clear the **Enable Quick Launch** check box. You can also enable or disable search categories on this page.

Category List

Quick Launch search results appear in four categories: **Most Recently Used**, **Menus**, **Options**, and **Open Documents**, along with the number of items in the category. To traverse through search results by category, choose the **Ctrl+Q** keys to show all the results from the next category. After the last category appears, **Ctrl+Q** shows you a few results from each category. Press **Ctrl+Shift+Q** to navigate through the categories in reverse order. To view all the search results under a category, choose the category name.

You can use the following shortcuts to limit your search to specific categories.

CATEGORY	SHORTCUT	SHORTCUT DESCRIPTION
Most recently used	@mru For example, <code>@mru font</code>	Displays up to five of the items that you Most Recently Used .
Menus	@menu For example, <code>@menu project</code>	Limits the search to menu items.
Options	@opt For example, <code>@opt font</code>	Limits the search to settings in the Options dialog box.
Documents	@doc For example, <code>@doc program.cs</code>	Limits the search to file names and paths of open documents for the search criteria, but doesn't search the text inside the files themselves.

NOTE

You can change the shortcut keys on the **General > Keyboard** page in the **Options** dialog box.

Show Previous Results

By default, the search term that you enter is not persisted between search sessions. The search string is cleared if you search for a term, move the cursor outside the **Quick Launch** area, and then go back. To retain the search results, go to the **Options** dialog box, choose **Quick Launch**, and then select the **Show search results from previous search when Quick Launch is activated** check box. The next time you do a search, leave the Quick

Launch area, and come back, Quick Launch will retain the search term last used and also show you the search results.

Options dialog box: Environment > Startup

10/18/2019 • 2 minutes to read • [Edit Online](#)

Use this page to customize the Visual Studio start page or set a different default action when Visual Studio starts up. For more information, see [Customize startup](#).

Use this page to set a different default action when Visual Studio starts up. In the **On startup, open** list, choose from **Start window** (which lets you open a new or existing project), **Most recent solution**, or **Empty environment**.

Options dialog box: Environment > Tabs and Windows

10/18/2019 • 2 minutes to read • [Edit Online](#)

Use this page to set options for how tabbed windows behave in the editor pane and how tool windows behave. For more information, see [Customize window layouts in Visual Studio](#)

Options dialog box: Environment > Task List

10/18/2019 • 2 minutes to read • [Edit Online](#)

This Options page allows you to add, delete, and change the comment tokens that generate **Task List** reminders. To display these settings, select **Options** from the **Tools** menu, expand the **Environment** folder, and choose **Task List**.

Task List Tokens

When you insert a comment into your code whose text begins with a token from the **Token list**, **Task List** displays your comment as new entry whenever the file is opened for editing. Click a **Task List** entry to jump directly to the comment line in your code. For more information, see [Using the Task List](#).

Token List

Displays a list of tokens, and allows you to add or remove custom tokens. Comment tokens are case sensitive in C# and C++, but not in Visual Basic.

NOTE

If you don't type the desired token exactly as it appears in the token list, a comment task will not be displayed in **Task List**.

Priority

Sets the priority of tasks that use the selected token (low, normal, or high). Task comments that begin with this token are automatically assigned the designated priority in **Task List**.

Name

Enter the token string here and then click **Add** to add the string to the token list.

Add

Enabled when you enter a new **Name**. Click to add a new token string using the values entered in the **Name** and **Priority** fields.

Delete

Click to delete the selected token from the token list. You cannot delete the default comment token.

Change

Click to make changes to an existing token using the values entered in the **Name** and **Priority** fields.

NOTE

You can't rename or delete the default comment token, but you can change its priority level.

See also

- [Use the Task List](#)
- [Set Bookmarks in Code](#)

Configure trust settings for files and folders

2/8/2019 • 2 minutes to read • [Edit Online](#)

Visual Studio prompts for user approval before opening projects that have the [mark of the web](#). For added security, you can also configure Visual Studio to prompt for user approval before opening any file or folder that has the mark of the web attribute, or that isn't designated as *trusted*. File and folder checks are disabled by default.

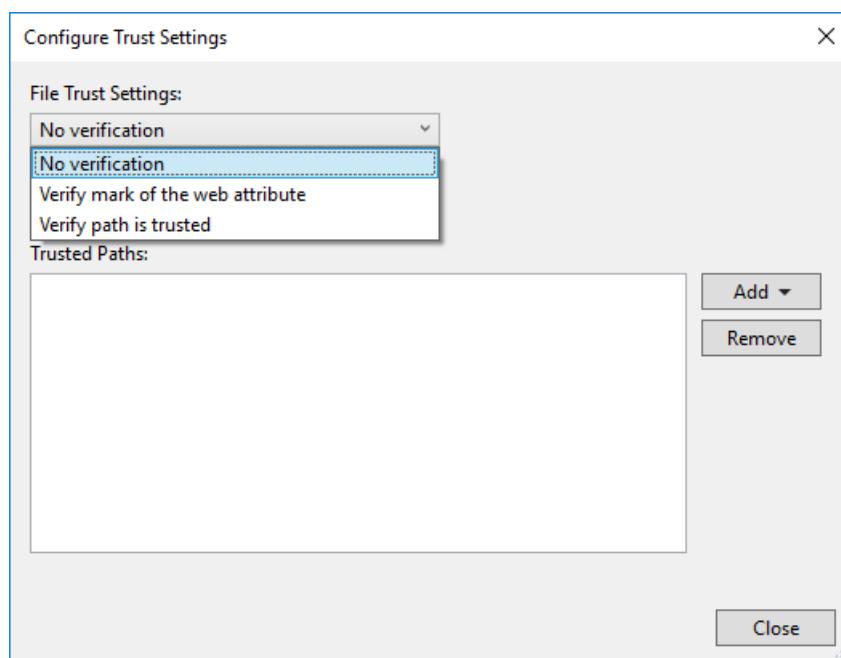
WARNING

You should still ensure that the file, folder, or solution comes from a trusted person or a trusted location before approving it.

Configure trust settings

To change trust settings, follow these steps:

1. Open **Tools > Options > Trust Settings** and select the **Configure Trust Settings** link in the right-hand pane.
2. Choose the level of checks you'd like for files and folders. You can have different checks for each one. The options are:
 - **No verification**: Visual Studio doesn't perform any checks.
 - **Verify mark of the web attribute**: If the file or folder has the mark of the web attribute, Visual Studio blocks and asks for permission to open.
 - **Verify path is trusted**: If the file or folder path isn't part of the **Trusted Paths** list, Visual Studio blocks and asks for permission to open.



Add trusted paths

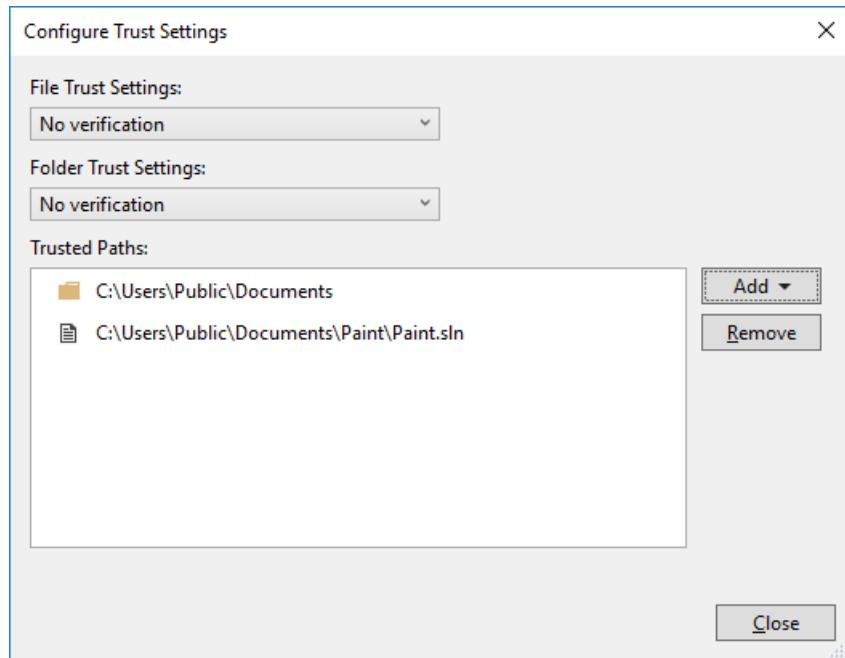
To add trusted paths, follow these steps:

1. Open **Tools > Options > Trust Settings** and select the **Configure Trust Settings** link in the right-hand

pane.

2. Click **Add** in the **Trust Settings** dialog, and then select **File or Folder**.
3. Navigate to and select the file or folder you want to add to the trusted list.

The file or folder path appears in the **Trusted Paths** list.



Remove trusted paths

To remove trusted paths, follow these steps:

1. Open **Tools > Options > Trust Settings** and select the **Configure Trust Settings** link in the right-hand pane.
2. Select the path you'd like to remove in the **Trusted Paths** list, and then click **Remove**.

TIP

To select multiple entries, hold down **Shift** while you select the paths.

The selected paths are removed from the **Trusted Paths** list.

Options dialog box: Environment > Web Browser

10/18/2019 • 2 minutes to read • [Edit Online](#)

Sets options for both the internal Web browser and Internet Explorer. To access this dialog box, click **Options** on the **Tools** menu, expand the **Environment** folder, and select **Web Browser**.

NOTE

The dialog boxes and menu commands you see might differ from those described in Help depending on your active settings or edition. To change your settings, choose **Import and Export Settings** on the **Tools** menu. For more information, see [Reset settings](#).

IMPORTANT

Opening certain files or components from the Web can execute code on your computer.

Home page

Sets the page displayed when you open the IDE Web Browser.

Search page

Lets you designate a Search page for the internal Web browser. This location can differ from the search page used by instances of Internet Explorer initiated outside of the integrated development environment (IDE).

View source in

Sets the editor used to open a Web page when you choose **View Source** on the page from the internal Web browser.

- **Source editor** Select to view source in the [editor](#).
- **HTML editor** Select to view source in the [HTML designer](#). Use this selection to edit the Web page in one of two views: Design view or the standard text-based Source view.
- **External editor** Select to view source in another editor. Specify the path of any editor you choose, for example, Notepad.exe.

Internet Explorer Options

Click to change options for Internet Explorer in the **Internet Properties** dialog box. Changes made in this dialog box affect both the internal Web browser and instances of Internet Explorer initiated outside of the Visual Studio IDE (for example, from the Start menu).

NOTE

Use the **Browse With** dialog box to replace the Visual Studio internal Web browser with a browser of your choice. You can access the Browse With dialog box from the right-click or context menu of, for example, an HTML file in your project.

See also

- [General, Environment, Options Dialog Box](#)
- [HTML Designer](#)

Options dialog box: Projects and Solutions > General

10/18/2019 • 4 minutes to read • [Edit Online](#)

Use this page to define Visual Studio's behavior related to projects and solutions. To access these options, select **Tools > Options**, expand **Projects and Solutions**, and then select **General**.

The following options are available on the **General** page.

Always show Error List if build finishes with errors

Opens the **Error List** window on build completion, only if a project failed to build. Errors that occur during the build process are displayed. When this option is cleared, the errors still occur but the window does not open when the build is complete. This option is enabled by default.

Track active item in Solution Explorer

When selected, **Solution Explorer** automatically opens and the active item is selected. The selected item changes as you work with different files in a project or solution, or different components in a designer. When this option is cleared, the selection in **Solution Explorer** does not change automatically. This option is enabled by default.

Show advanced build configurations

When selected, the build configuration options appear on the **Project Property Pages** dialog box and the **Solution Property Pages** dialog box. When cleared, the build configuration options do not appear on the **Project Property Pages** dialog box and the **Solution Property Pages** dialog box for Visual Basic and C# projects that contain one configuration or the two configurations debug and release. If a project has a user-defined configuration, the build configuration options are shown.

When unselected, the commands on the **Build** menu, such as **Build Solution**, **Rebuild Solution**, and **Clean Solution**, are performed on the Release configuration and the commands on the **Debug** menu, such as **Start Debugging** and **Start Without Debugging**, are performed on the Debug configuration.

Always show solution

When selected, the solution and all commands that act on solutions are always shown in the IDE. When cleared, all projects are created as stand-alone projects and you do not see the solution in Solution Explorer or commands that act on solutions in the IDE if the solution contains only one project.

Save new projects when created

When selected, you can specify a location for your project in the **New Project** dialog box. When cleared, all new projects are created as temporary projects. When you are working with temporary projects, you can create and experiment with a project without having to specify a disk location.

Warn user when the project location is not trusted

If you attempt to create a new project or open an existing project in a location that is not fully trusted (for example, on a UNC path or an HTTP path), a message is displayed. Use this option to specify whether the message is displayed each time that you attempt to create or open a project in a location that is not fully trusted.

Show Output window when build starts

Automatically displays the [Output window](#) in the IDE at the outset of solution builds.

Prompt for symbolic renaming when renaming files

When selected, Visual Studio displays a message box asking whether or not it should also rename all references in the project to the code element.

Prompt before moving files to a new location

When selected, Visual Studio displays a confirmation message box before the locations of files are changed by actions in **Solution Explorer**.

Reopen documents on solution load

When selected, documents that were left open the previous time the solution was closed are automatically opened when the solution is opened.

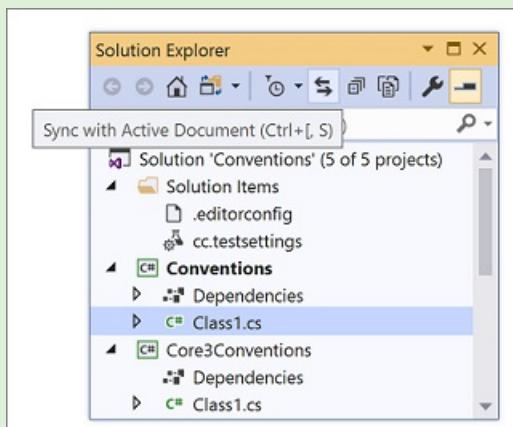
Reopening certain types of files or designers can delay solution load. Uncheck this option to [improve solution load performance](#) if you don't want to restore the solution's previous context.

Restore Solution Explorer project hierarchy state on solution load

When selected, restores the state of nodes in Solution Explorer with respect to whether they were expanded or collapsed the last time the solution was open. Deselect this option to decrease solution load time for large solutions.

TIP

If you disable this option, an easy way to navigate to the active document in Solution Explorer is by selecting **Sync with Active Document** on the **Solution Explorer** toolbar.



Open SDK-style project files with double-click or the Enter key

When this option is selected and you double-click on an SDK-style project node in Solution Explorer or select it and then press **Enter**, the project file (for example, *.csproj file) opens as XML in the editor. When deselected, double-clicking an SDK-style project node in Solution Explorer or selecting it and pressing **Enter** has the effect of expanding or collapsing the node only.

If you don't have this option selected and you want to edit an SDK-style project file, right-click on the project node in Solution Explorer and select **Edit Project File**. For other project types, you must first unload the project before

editing it in Visual Studio.

TIP

An *SDK-style project*, or [project SDK](#), has a newer, more streamlined project file format that was introduced with MSBuild 15.0. An SDK-style project contains an `Sdk` attribute on the `Project` element, for example

```
<Project Sdk="Microsoft.NET.Sdk">
```

Visual Studio creates an SDK-style project when you create a new .NET Core project from one of the Visual Studio templates, for example.

See also

- [Options dialog box: Projects and Solutions > Locations](#)
- [Options Dialog Box, Projects and Solutions, Build and Run](#)
- [Options Dialog Box, Projects and Solutions, Web Projects](#)

Options dialog box: Projects and Solutions > Build and Run

7/24/2019 • 2 minutes to read • [Edit Online](#)

In this dialog box, you can specify the maximum number of C++ or C# projects that can build at the same time, certain default build behaviors, and some build log settings. To access these options, select **Tools** > **Options**, expand **Projects and Solutions**, and then select **Build and Run**.

Maximum number of parallel project builds

Specifies the maximum number of C++ and C# projects that can build at the same time. To optimize the build process, the maximum number of parallel project builds is automatically set to the number of CPUs of your computer. The maximum is 32.

Only build startup projects and dependencies on Run

Builds only the startup project and its dependencies when you use the **F5** key, the **Debug** > **Start Debugging** menu command, or applicable commands on the **Build** menu. If unchecked, all projects and dependencies are built.

On Run, when projects are out of date

Applies to C++ projects only.

When running a project with **F5** or the **Debug** > **Start Debugging** command, the default setting **Prompt to build** displays a message if a project configuration is out of date. Select **Always build** to build the project every time it is run. Select **Never build** to suppress all automatic builds when a project is run.

On Run, when build or deployment errors occur

Applies to C++ projects only.

When running a project with **F5** or the **Debug** > **Start Debugging** command, the default setting **Prompt to launch** displays a message if a project should be run even if the build failed. Select **Launch old version** to automatically launch the last good build, which could result in mismatches between the running code and the source code. Select **Do not launch** to suppress the message.

For new solutions use the currently selected project as the startup project

When this option is set, new solutions use the currently selected project as the startup project.

MSBuild project build output verbosity

Determines how much information from the build process is displayed in the **Output** window.

MSBuild project build log file verbosity

Applies to C++ projects only.

Determines how much information is written to the build log file, which is located at `\<ProjectName>\Debug\<ProjectName>.log`.

See also

- [Compiling and Building](#)

- [Options Dialog Box, Projects and Solutions](#)
- [Options Dialog Box, Projects and Solutions, Web Projects](#)

Options dialog box: Projects and Solutions > Locations

10/18/2019 • 2 minutes to read • [Edit Online](#)

Use this page to set default locations for projects, project templates, and item templates. To access these options, select **Tools** > **Options**, expand **Projects and Solutions**, and then select **Locations**.

The following options are available on the **Locations** page.

Projects location

Specifies the default location where Visual Studio creates new projects and solution folders. Several dialog boxes also use the location set in this option for folder starting points. For example, the **Open Project** dialog box uses this location for the **My Projects** shortcut.

User project templates location

Specifies the default location that's used to search for user-created project templates. For more information, see [How to: Locate and Organize Templates](#).

User item templates location

Specifies the default location that's used to search for user-created item templates. For more information, see [How to: Locate and Organize Templates](#).

See also

- [Options dialog box: Projects and Solutions > General](#)
- [Options Dialog Box, Projects and Solutions, Build and Run](#)
- [Options Dialog Box, Projects and Solutions, Web Projects](#)

Visual Basic Defaults, Projects, Options Dialog Box

10/21/2019 • 2 minutes to read • [Edit Online](#)

Specifies the default settings for Visual Basic project options. When a new project is created, the specified option statements will be added to the project header in the Code Editor. The options apply to all Visual Basic projects.

To access this dialog box, on the **Tools** menu, click **Options**, expand the **Projects and Solutions** folder, and then click **VB Defaults**.

Option Explicit

Sets the compiler default so that explicit declarations of variables are required. By default, **Option Explicit** is set to **On**. For more information, see [/optionexplicit](#).

Option Strict

Sets the compiler default so that explicit narrowing conversions are required and late binding is not allowed. By default, **Option Strict** is set to **Off**. For more information, see [/optionstrict](#).

Option Compare

Sets the compiler default for string comparisons: binary (case-sensitive) or text (case-insensitive.) By default, **Option Compare** is set to **Binary**. For more information, see [/optioncompare](#).

Option Infer

Sets the compiler default for local type inference. By default, **Option Infer** is set to **On** for newly created projects and to **Off** for migrated projects created in earlier versions of Visual Basic. For more information, see [/optioninfer](#).

See also

- [Solutions and Projects](#)

VC++ Project Settings, Projects and Solutions, Options Dialog Box

8/9/2019 • 2 minutes to read • [Edit Online](#)

This dialog box lets you define C++ build and project settings related to logging, performance, and supporting file types.

To access this dialog box

1. On the **Tools** menu, click **Options**.
2. Select **Projects and Solutions**, and then select **VC++ Project Settings**.

Build Logging

Yes

Turns on generation of the build log file. This option generates BuildLog.htm, which can be found in the project's intermediate files directory. Every fresh build overwrites the previous BuildLog.htm file.

No

Turns off generation of the build log file.

Show Environment in Log

Yes

Lists environment variables in the build log file. This option specifies to echo all environment variables, during builds of C++ projects, into the build log file.

No

Exclude environment variables from the build log file.

Build Timing

Yes

Turns on build timing. If selected, the time it takes for the build to complete is posted to the Output window. For more information, see [Output Window](#).

No

Turns off build timing.

Maximum concurrent C++ compilations

Specifies the maximum number of CPU cores to use for parallel C++ compilation.

Extensions to Include

Specifies the file name extensions of files that can be ported into your project.

Extensions to Hide

Specifies the file name extensions of files that will not be displayed in **Solution Explorer** when **Show All Files** is enabled.

Build Customization Search Path

Specifies the list of directories that contain .rules files, which help you define build rules for your projects.

Solution Explorer Mode

Show only files in project

Configures **Solution Explorer** to only display files in the project.

Show all files

Configures **Solution Explorer** to show files in the project and files on disk in the project folder.

Enable Project Caching

Yes

Enables Visual Studio to cache project data so that when you open the project the next time, it can load that cached data rather than recomputing it from the project files. Using cached data can speed up the project load time significantly.

No

Do not use cached project data. Parse the project files each time the project loads.

See also

- [Building C/C++ Programs](#)
- [C/C++ Building Reference](#)

Options Dialog Box, Projects and Solutions, Web Projects

10/25/2019 • 2 minutes to read • [Edit Online](#)

Sets the web server that web projects will use for development within Visual Studio. To access these options, select **Tools > Options** expand **Projects and Solutions**, and select **Web Projects**.

By default, running a web project in Visual Studio uses the Visual Studio Development Server. For more information, see [Web Servers in Visual Studio for ASP.NET Web Projects](#).

Settings

Use the 64-bit version of IIS Express for web sites and projects

Select this option to use IIS Express instead of the Visual Studio Development Server. For more information, see [Introducing IIS Express](#) and [IIS Express Overview](#).

Warn before running web applications when there are errors in the error list

If this option is set, you will be warned if you try to run your web application when it does not compile without errors.

See also

- [Options Dialog Box, Projects and Solutions](#)
- [Options Dialog Box, Projects and Solutions, Build and Run](#)

Options dialog box: Text Editor > General

10/18/2019 • 2 minutes to read • [Edit Online](#)

This dialog box lets you change global settings for the Visual Studio code and text editor. To display this dialog box, select **Options** on the **Tools** menu, expand the **Text Editor** folder, and then select **General**.

Settings

Drag and drop text editing

When selected, enables you to move text by selecting it and dragging it with the mouse to another location within the current document or any other open document.

Automatic delimiter highlighting

When selected, delimiter characters that separate parameters or item-value pairs, as well as matching braces, are highlighted.

Track changes

When the code editor is selected, a vertical yellow line appears in the selection margin to mark code that has changed since the file was most recently saved. When you save the changes, the vertical lines become green.

Auto-detect UTF-8 encoding without signature

By default, the editor detects encoding by searching for byte order marks or charset tags. If neither is found in the current document, the code editor tries to autodetect UTF-8 encoding by scanning byte sequences. To disable the autodetection of encoding, clear this option.

Follow project coding conventions

When selected, the project's specified coding conventions override any coding conventions you use on your personal projects.

Enable mouse-click to perform Go to Definition

When selected, you can press **Ctrl** and hover over an element while clicking the mouse. Doing so takes you to the definition of the selected element. You can also choose either **Alt** or **Ctrl + Alt** from the **Use modifier key** dropdown.

Select the **Open definition in peek view** check box to display the element's definition in a window without navigating away from your current location in the code editor.

Display

Selection margin

When selected, displays a vertical margin along the left edge of the editor's text area. You can click this margin to select an entire line of text, or click and drag to select consecutive lines of text.



Indicator margin

When selected, displays a vertical margin outside the left edge of the editor's text area. When you click in this

margin, an icon and ToolTip that are related to the text appear. For example, breakpoint or task list shortcuts appear in the indicator margin. Indicator Margin information doesn't print.

Highlight current line

When selected, displays a gray box around the line of code in which the cursor is located.

Show structure guide lines

When selected, vertical lines appear in the editor that line up with structured code blocks, which lets you easily identify the individual blocks of code.

See also

- [Options, Text Editor, All Languages](#)
- [Options, Text Editor, All Languages, Tabs](#)
- [Options, Text Editor, File Extension](#)
- [Identifying and Customizing Keyboard Shortcuts](#)
- [Customizing the Editor](#)
- [Using IntelliSense](#)

Options, Text Editor, File Extension

10/18/2019 • 2 minutes to read • [Edit Online](#)

This Options dialog allows you to specify how all files with certain file extensions will be handled by the Visual Studio integrated development environment (IDE). For each **Extension** that you enter, you can select an Editing Experience. This allows you to choose the IDE editor or designer in which documents of a certain type will open. To display these options, choose **Options** from the **Tools** menu, expand the **Text Editor** node, and select **File Extension**.

When you select an option "with Encoding," a dialog will be displayed whenever you open a document of that type that allows you to select an encoding scheme for that document. This can be helpful if you are preparing versions of your project documents for use on different platforms or in different target languages.

UIElement list

Extension

Type the file extension whose Editing Experience in the IDE you wish to define.

Editor

Select the IDE editor or designer in which documents with this file extension will open. When you select an option "with Encoding," a dialog will be displayed whenever you open such a document that allows you to select an encoding scheme.

Add

Adds an entry that includes the specified **Extension** and **Editing Experience** to the Extension List.

Remove

Deletes the selected entry from the Extension List.

Extension List

Lists all extensions for which an Editing Experience has been specified.

Map extensionless files to

Select this option if you wish to specify how files without an extension will be handled by the IDE.

Extensionless file options

Provides the same list as **Editor**. Select the IDE editor or designer in which documents without file extensions will open.

See also

- [How to: Manage Editor Modes](#)

Options dialog box: Text Editor > All Languages

10/21/2019 • 3 minutes to read • [Edit Online](#)

This dialog box allows you to change the default behavior of the Code Editor. These settings also apply to other editors based upon the Code Editor, such as the HTML Designer's Source view. To open this dialog box, select **Options** from the **Tools** menu. Within the **Text Editor** folder, expand the **All Languages** subfolder and then choose **General**.

Caution

This page sets default options for all development languages. Remember that resetting an option in this dialog will reset the General options in all languages to whatever choices are selected here. To change Text Editor options for just one language, expand the subfolder for that language and select its option pages.

A grayed checkmark is displayed when an option has been selected on the General options pages for some programming languages, but not for others.

Statement Completion

Auto list members

When selected, pop-up lists of available members, properties, values, or methods are displayed by IntelliSense as you type in the editor. Choose any item from the pop-up list to insert the item into your code. Selecting this option enables the **Hide advanced members** option.

Hide advanced members

When selected, shortens pop-up statement completion lists by displaying only those items most commonly used. Other items are filtered from the list.

Parameter information

When selected, the complete syntax for the current declaration or procedure is displayed under the insertion point in the editor, with all of its available parameters. The next parameter you can assign is displayed in bold.

Settings

Enable virtual space

When this option is selected and **Word wrap** is cleared, you can click anywhere beyond the end of a line in the Code Editor and type. This feature can be used to position comments at a consistent point next to your code.

Word wrap

When selected, any portion of a line that extends horizontally beyond the viewable editor area is automatically displayed on the next line. Selecting this option enables the **Show visual glyphs for word wrap** option.

NOTE

The **Virtual Space** feature is turned off while **Word Wrap** is on.

Show visual glyphs for word wrap

When selected, a return-arrow indicator is displayed where a long line wraps onto a second line.



Clear this option if you prefer not to display these indicators.

NOTE

These reminder arrows are not added to your code, and do not print. They are for reference only.

Line numbers

When selected, a line number appears next to each line of code.

NOTE

These line numbers are not added to your code, and do not print. They are for reference only.

Enable single-click URL navigation

When selected, the mouse cursor changes to a pointing hand as it passes over a URL in the editor. You can click the URL to display the indicated page in your web browser.

Navigation bar

When selected, displays the **Navigation bar** at the top of the code editor. Its dropdown **Objects** and **Members** lists allow you to choose a particular object in your code, select from its members, and navigate to the declaration of the selected member in the Code Editor.

Apply Cut or Copy commands to blank lines when there is no selection

This option sets the behavior of the editor when you place the insertion point on a blank line, select nothing, and then Copy or Cut.

- When this option is selected, the blank line is copied or cut. If you then Paste, a new, blank line is inserted.
- When this option is cleared, the Cut command removes blank lines. However, the data on the Clipboard is preserved. Therefore, if you then use the Paste command, the content most recently copied onto the Clipboard is pasted. If nothing has been copied previously, nothing is pasted.

This setting has no effect on Copy or Cut when a line is not blank. If nothing is selected, the entire line is copied or cut. If you then Paste, the text of the entire line and its endline character are pasted.

TIP

To display indicators for spaces, tabs, and line ends, and thus distinguish indented lines from lines that are entirely blank, select **Advanced** from the **Edit** menu and choose **View White Space**.

See also

- [Options, Text Editor, All Languages, Tabs](#)
- [General, Environment, Options Dialog Box](#)
- [Using IntelliSense](#)

Options, Text Editor, All Languages, Scroll Bars

10/18/2019 • 2 minutes to read • [Edit Online](#)

This dialog box lets you change the default behavior of the code editor scroll bar. To display these options, select **Options** from the **Tools** menu. Within the **Text Editor** folder, expand the **All Languages** subfolder, and then choose **Scroll Bars**.

Caution

This page sets default options for all development languages. Resetting an option in this dialog will reset the Scroll Bars options in all languages to whatever choices are selected here. To change Text Editor options for just one language, expand the subfolder for that language and select its option pages.

Show horizontal scroll bar

When selected, displays a horizontal scrollbar, which allows you to scroll from side-to-side to view elements that fall outside the viewing area of the Editor. If horizontal scrollbars are unavailable, you can use the cursor keys to scroll.

Show vertical scroll bar

When selected, displays a vertical scrollbar, which allows you to scroll up and down to view elements that fall outside the viewing area of the Editor. If vertical scrollbars are not available, you can use the Page Up, Page Down and cursor keys to scroll.

Display

Show annotations over vertical scroll bar

Select whether the vertical scroll bar shows the following annotations:

- changes
- marks
- errors
- caret position

TIP

The **Show marks** option includes breakpoints and bookmarks.

Try it out by opening a large code file and replacing some text that occurs in several places in the file. The scroll bar shows you the effect of the replacements, so you can back out your changes if you replaced something you shouldn't have.

See the [Enhanced scroll bar](#) blog post on what various colors and symbols mean when editing code.

Behavior

The scroll bar has two modes: bar mode and map mode.

Use bar mode for vertical scroll bar

Bar mode displays annotation indicators on the scroll bar. Clicking on the scroll bar scrolls the page up or down but

does not jump to that location in the file.

Use map mode for vertical scroll bar

In *map mode*, when you click a location on the scroll bar, the cursor jumps to that location in the file instead of just scrolling up or down a page. Lines of code are shown, in miniature, on the scroll bar. You can choose how wide the map column is by selecting a value in **Source overview**. To enable a larger preview of the code when you rest the pointer on the map, select the **Show Preview Tooltip** option. Collapsed regions are shaded differently and expand when you double-click them.

TIP

You can turn the miniature code view off in map mode by setting **Source overview** to **Off**. If **Show Preview Tooltip** is selected, you still see a preview of the code at that location when you hover your pointer on the scroll bar, and the cursor still jumps to that location in the file when you click.

See also

- [How to: Customize the scroll bar](#)

Options, Text Editor, All Languages, Tabs

10/18/2019 • 2 minutes to read • [Edit Online](#)

This dialog box allows you to change the default behavior of the Code Editor. These settings also apply to other editors based upon the Code Editor, such as the HTML Designer's Source view. To display these options, select **Options** from the **Tools** menu. Within the **Text Editor** folder expand the **All Languages** subfolder, and then choose **Tabs**.

Caution

This page sets default options for all development languages. Remember that resetting an option in this dialog will reset the Tabs options in all languages to whatever choices are selected here. To change Text Editor options for just one language, expand the subfolder for that language and select its option pages.

If different settings are selected on the Tabs options pages for particular programming languages, then the message "The indentation settings for individual text formats conflict with each other," is displayed for differing **Indenting** options; and the message "The tab settings for individual text formats conflict with each other," is displayed for differing **Tab** options. For example, this reminder is displayed if the **Smart indenting** option is selected for Visual Basic, but **Block indenting** is selected for Visual C++.

Indenting

None

When selected, new lines are not indented. The insertion point is placed in the first column of a new line.

Block

When selected, new lines are automatically indented. The insertion point is placed at the same starting point as the preceding line.

Smart

When selected, new lines are positioned to fit the code context, per other code formatting settings and IntelliSense conventions for your development language. This option is not available for all development languages.

For example, lines enclosed between an opening brace ({) and a closing brace (}) might automatically be indented an extra tab stop from the position of the aligned braces.

Tabs

Tab size

Sets the distance in spaces between tab stops. The default is four spaces.

Indent size

Sets the size in spaces of an automatic indentation. The default is four spaces. Tab characters, space characters, or both will be inserted to fill the specified size.

Insert spaces

When selected, indent operations insert only space characters, not TAB characters. If the **Indent size** is set to 5, for example, then five space characters are inserted whenever you press the TAB key or the **Increase Indent** button on the **Formatting** toolbar.

Keep tabs

When selected, indent operations insert as many TAB characters as possible. Each TAB character fills the number of spaces specified in **Tab size**. If the **Indent size** is not an even multiple of the **Tab size**, space characters are added to fill in the difference.

See also

- [Options, Text Editor, All Languages](#)
- [General, Environment, Options Dialog Box](#)

Options, Text Editor, Basic (Visual Basic), Advanced

10/18/2019 • 3 minutes to read • [Edit Online](#)

The **VB Specific** property page, in the **Basic** folder of the **Text Editor** folder of the **Options (Tools** menu) dialog box includes the following properties:

Analysis

- Enable full solution analysis

Enables code analysis on all files in the solution, not just open code files. For more information, see [Full solution analysis](#).

Using Directives

- Place 'System' directives first when sorting usings

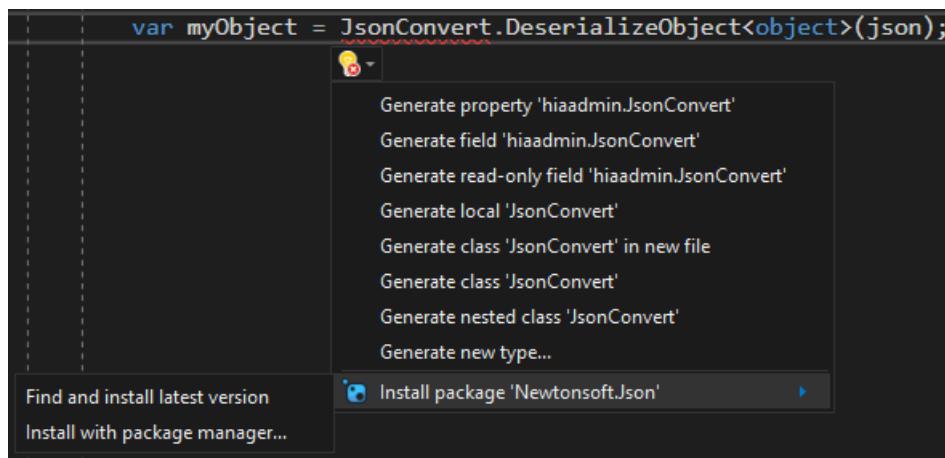
When selected, the **Remove and Sort Usings** command in the right-click menu sorts the `using` directives and places the 'System' namespaces at the top of the list.

- Separate using directive groups

When selected, the **Remove and Sort Usings** command in the right-click menu separates `using` directives by inserting an empty line between groups of directives that have the same root namespace.

- Suggest usings for types in reference assemblies
- Suggest usings for types in NuGet packages

When these options are selected, a [Quick Action](#) is available to install a NuGet package and add a `using` directive for unreferenced types.



Highlighting

Enable highlighting of references and keywords

The text editor can highlight all instances of a symbol or all keywords in a clause such as `If..Then`, `While...End While`, or `Try...Catch...Finally`. You can navigate between highlighted references or keywords by pressing **Ctrl + Shift + Down arrow** or **Ctrl + Shift + Up arrow**.

Outlining

Enable outlining mode

When you open a file in the code editor, you can view the document in outlining mode. See [Outlining](#) for more information. When this option is selected, the outlining feature is activated when you open a file.

Show procedure line separators

The text editor indicates visual scope of procedures. A line is drawn in the .vb source files of your project at locations listed in the following table:

LOCATION IN .VB SOURCE FILE	EXAMPLE OF LINE LOCATION
After the close of a block declaration construct	- At the end of a class, structure, module, interface, or enum - After a property, function, or sub - Not between the get and set clauses in a property
After a set of single line constructs	- After the import statements, before a type definition in a class file - After variables declared in a class, before any procedures
After single line declarations (non-block level declarations)	- Following import statements, inherits statements, variable declarations, event declarations, delegate declarations, and DLL declare statements

Block Structure Guides

When selected, vertical lines appear in the editor that line up with structured code blocks, which lets you easily identify the individual blocks of code. For example, you would see a line between `Sub` and `EndSub` in a `Sub` statement.

Editor Help

Pretty Listing (reformatting) of code The text editor reformats your code as appropriate. When this option is selected, the code editor will:

- Align your code to the correct tab position
- Recase keywords, variables, and objects to the correct case
- Add a missing `Then` to an `If...Then` statement
- Add parenthesis to function calls
- Add missing end quotes to strings
- Reformat exponential notation
- Reformat dates

Automatic insertion of end constructs

When you type—for example, the first line of a procedure declaration, `Sub Main`—and press **Enter**, the text editor adds a matching `End Sub` line. Similarly, if you add a `For` loop, the text editor adds a matching `Next` statement. When this option is selected, the code editor automatically adds the end construct.

Automatic insertion of Interface and MustOverride members

When you commit an `Implements` statement or an `Inherits` statement for a class, the text editor inserts prototypes for the members that have to be implemented or overridden, respectively.

Enable error correction suggestions

The text editor can suggest solutions to common errors and allow you to select the appropriate correction, which is then applied to your code.

See also

- [General, Environment, Options Dialog Box](#)
- [Options, Text Editor, All Languages, Tabs](#)

Code style preferences

10/18/2019 • 4 minutes to read • [Edit Online](#)

You can define code style settings per-project by using an [EditorConfig file](#), or for all code you edit in Visual Studio on the text editor [Options page](#). For C# code, you can also configure Visual Studio to apply these code style preferences using the **Code Cleanup** (Visual Studio 2019) and **Format Document** (Visual Studio 2017) commands.

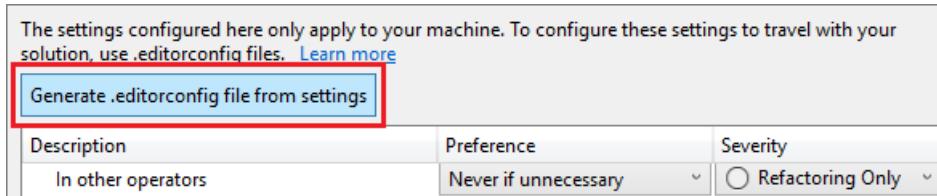
NOTE

This topic applies to Visual Studio on Windows. For Visual Studio for Mac, see [Editor behavior in Visual Studio for Mac](#).

Code styles in EditorConfig files

[Code style settings](#) for .NET can be specified by adding an [EditorConfig](#) file to your project. EditorConfig files are associated with a codebase rather than a Visual Studio personalization account. Settings in an EditorConfig file take precedence over code styles that are specified in the **Options** dialog box. Use an EditorConfig file when you want to enforce coding styles for all contributors to your repo or project.

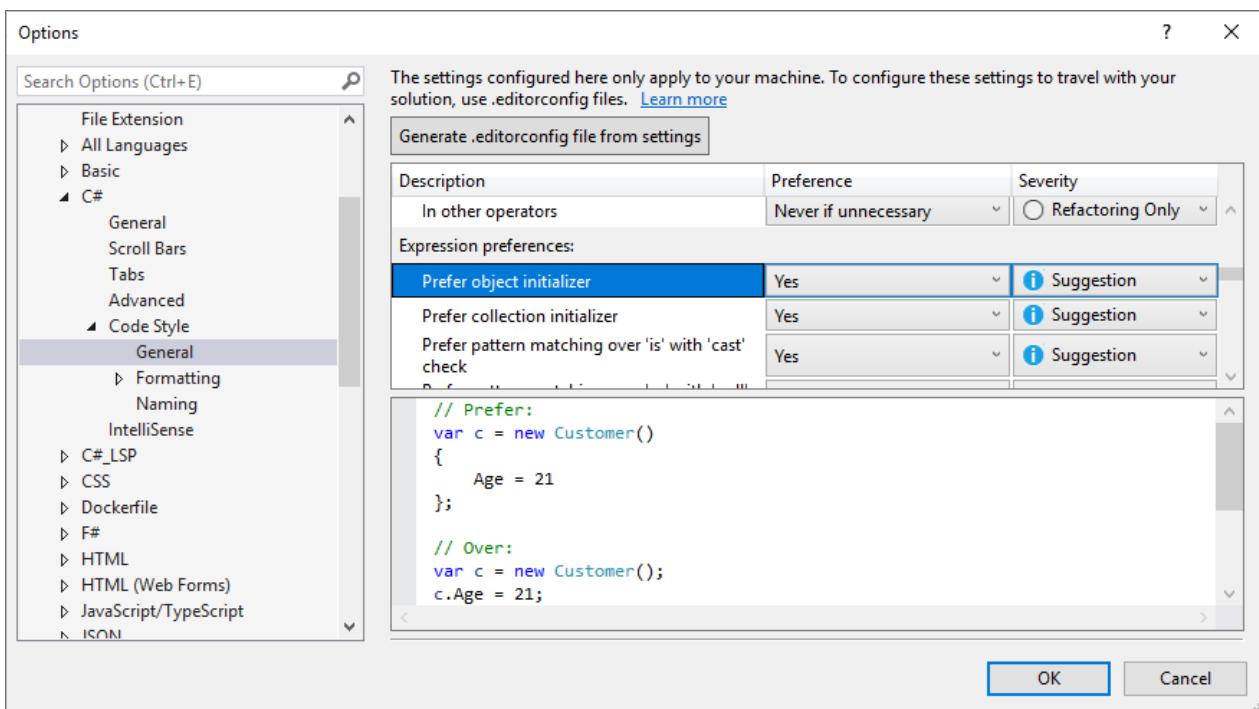
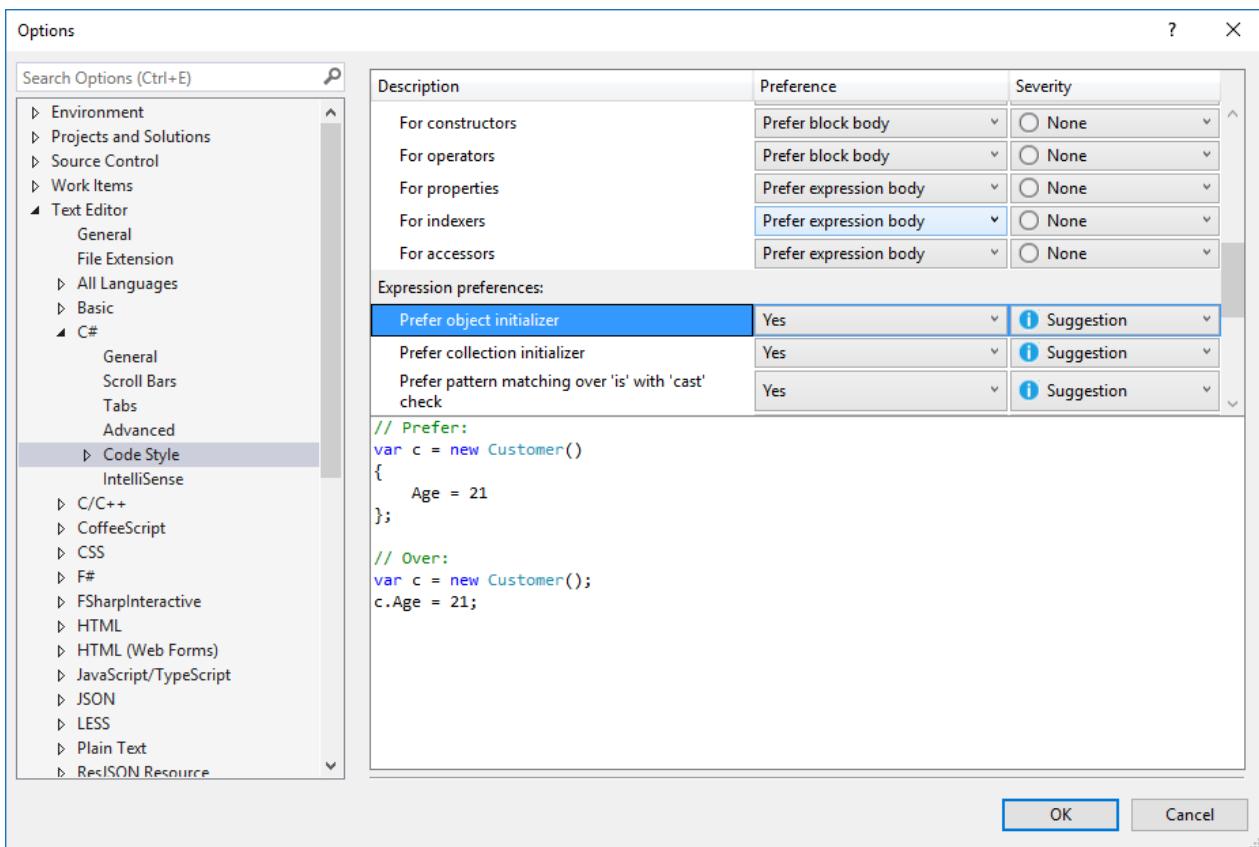
You can manually populate your EditorConfig file, or you can automatically generate the file based on the code style settings you've chosen in the Visual Studio **Options** dialog box. This options page is available at **Tools** > **Options** > **Text Editor** > **[C# or Basic]** > **Code Style** > **General**. Click **Generate .editorconfig file from settings** to automatically generate a coding style `.editorconfig` file based on the settings on this **Options** page.



Code styles in the Options dialog box

Code style preferences can be set for all of your C# and Visual Basic projects by opening the **Options** dialog box from the **Tools** menu. In the **Options** dialog box, select **Text Editor** > **[C# or Basic]** > **Code Style** > **General**.

Each item in the list shows a preview of the preference when selected:



Options set in this window are applicable to your Visual Studio personalization account and aren't associated with a particular project or codebase. In addition, they aren't enforced at build time, including in continuous integration (CI) builds. If you want to associate code style preferences with your project and have the styles enforced during build, specify the preferences in an [.editorconfig file](#) that's associated with the project.

Preference and severity

For each code style setting on this page, you can set the **Preference** and **Severity** values using the drop-downs on each line. Severity can be set to **Refactoring Only**, **Suggestion**, **Warning**, or **Error**. If you want to enable **Quick Actions** for a code style, ensure that the **Severity** setting is set to something other than **Refactoring Only**. The **Quick Actions** light bulb , error light bulb , or screwdriver icon appears when a non-preferred style is used, and you can choose an option on the **Quick Actions** list to automatically rewrite code to the preferred style.

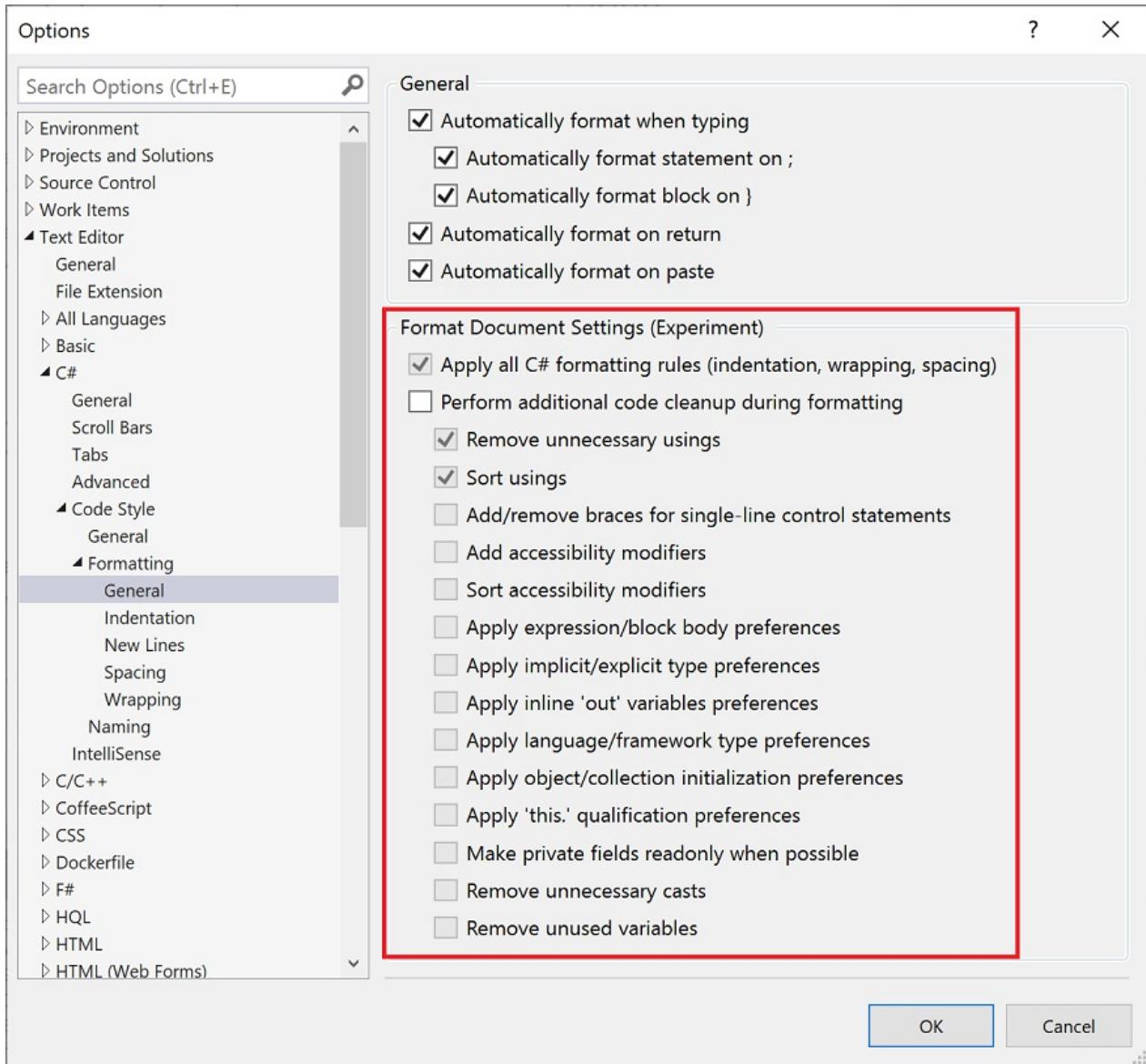
Apply code styles

You can configure the **Format Document** command (**Edit > Advanced > Format Document**) to apply your code style settings (from an EditorConfig file or **Code Style** options) along with the regular formatting that it does (such as indentation). If an `.editorconfig` file exists for the project, those settings take precedence.

NOTE

Applying code styles by using the **Format Document** command is only available for C# code files. This is an experimental feature.

Configure which settings you want **Format Document** to apply on the [Formatting options page](#).



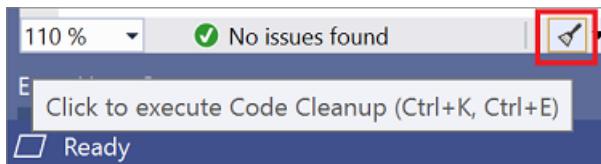
TIP

Rules configured with a severity of **None** don't participate in code cleanup but can be individually applied via the **Quick Actions and Refactorings** menu.

The first time you trigger the **Format Document** command, a yellow info bar prompts you to configure your code cleanup settings.

For C# code files, Visual Studio 2019 has a **Code Cleanup** button at the bottom of the editor (keyboard: **Ctrl+K**,

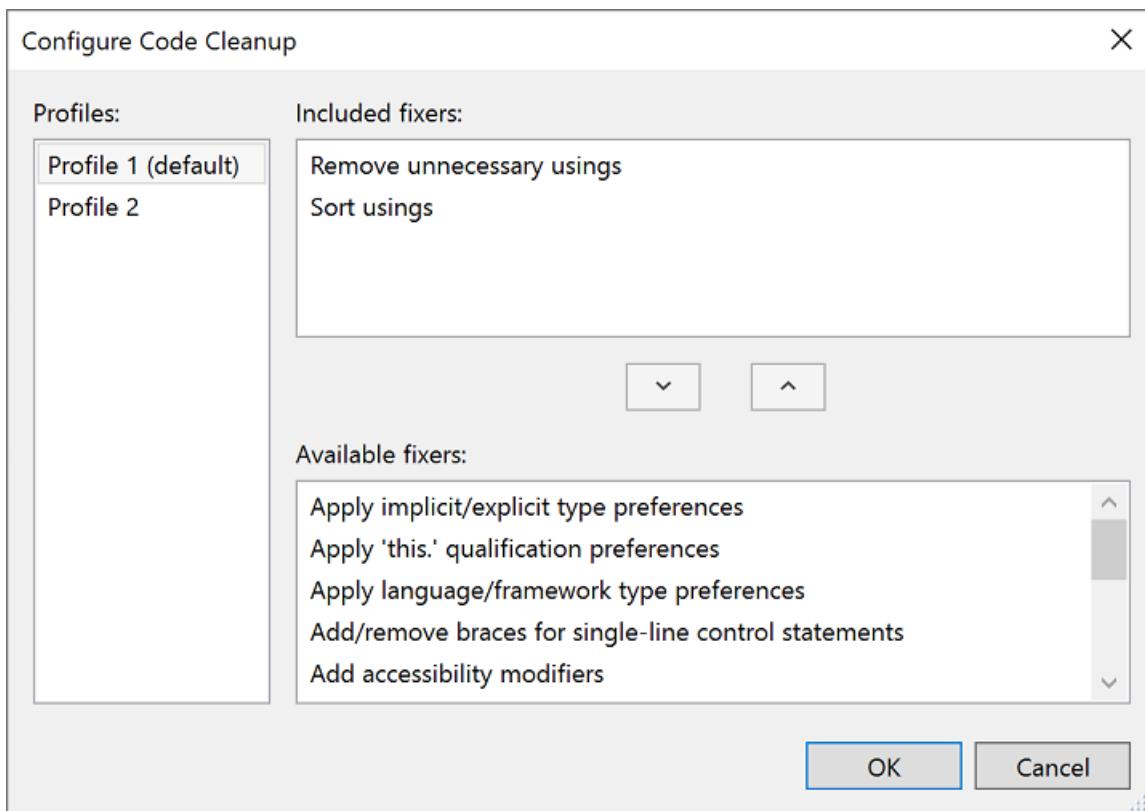
Ctrl+E) to apply code styles from an EditorConfig file or from the **Code Style** options page. If an *.editorconfig* file exists for the project, those are the settings that take precedence.



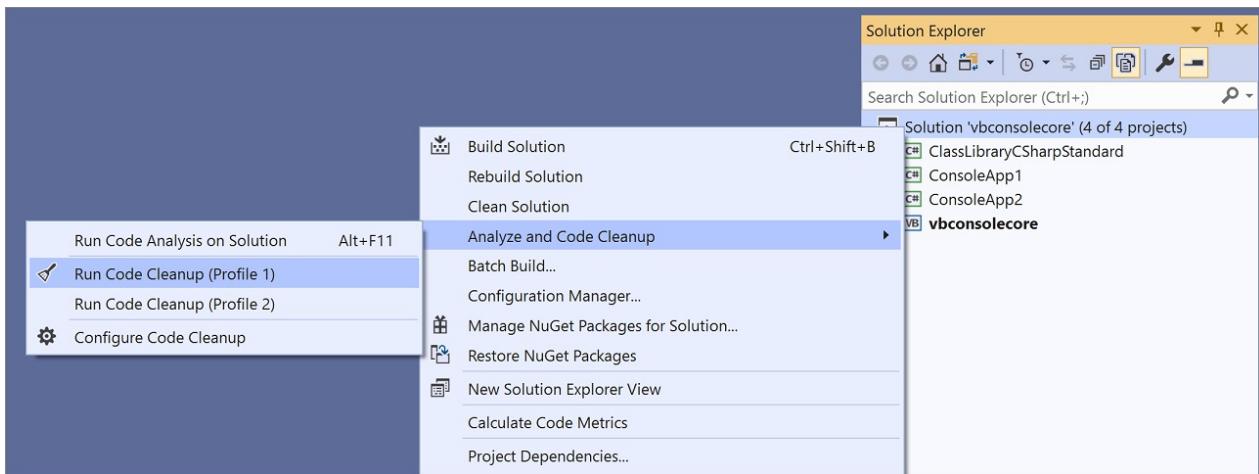
TIP

Rules configured with a severity of **None** don't participate in code cleanup but can be individually applied via the **Quick Actions and Refactorings** menu.

First, configure which code styles you want to apply (in one of two profiles) in the **Configure Code Cleanup** dialog box. To open this dialog box, click the expander arrow next to the code cleanup broom icon and then choose **Configure Code Cleanup**.



After you've configured code cleanup, you can either click on the broom icon or press **Ctrl+K, Ctrl+E** to run code cleanup. You can also run code cleanup across your entire project or solution. Right-click on the project or solution name in **Solution Explorer**, select **Analyze and Code Cleanup**, and then select **Run Code Cleanup**.



If you want your code style settings to be applied every time you save a file, you may like the [Code Cleanup on Save](#) extension.

See also

- [Quick Actions](#)
- [.NET coding convention settings for EditorConfig](#)
- [Editor behavior \(Visual Studio for Mac\)](#)

IntelliSense for Visual Basic code files

10/18/2019 • 2 minutes to read • [Edit Online](#)

The Visual Basic source code editor offers the following IntelliSense features:

Syntax tips

Syntax tips display the syntax of the statement that you are typing. This is useful for statements such as [Declare](#).

Automatic completion

- Completion on various keywords

For example, if you type `goto` and a space, IntelliSense displays a list of the defined labels in a drop-down menu. Other supported keywords include `Exit`, `Implements`, `Option`, and `Declare`.

- Completion on `Enum` and `Boolean`

When a statement will refer to a member of an enumeration, IntelliSense displays a list of the members of the `Enum`. When a statement will refer to a `Boolean`, IntelliSense displays a true-false drop-down menu.

Completion can be turned off by default by deselecting **Auto list members** from the **General** property page in the **Visual Basic** folder.

You can manually invoke completion by invoking List Members, Complete Word, or **Alt+Right Arrow**. For more information, see [Use IntelliSense](#).

IntelliSense in Zone

IntelliSense in Zone assists Visual Basic developers who need to deploy applications through ClickOnce and are constrained to partial trust settings. This feature:

- Enables you to choose the permissions the application will run with.
- Display APIs in the chosen Zone as available in List Members, and display APIs that require additional permissions as unavailable.

For more information, see [Code access security for ClickOnce applications](#).

Filtered completion lists

In Visual Basic, IntelliSense completion lists have two tab controls located near the bottom of the lists. The **Common** tab, which is selected by default, displays items that are most often used to complete the statement that you are writing. The **All** tab displays all items that are available for automatic completion, including those that are also on the **Common** tab.

See also

- [Use IntelliSense](#)

Options, Text Editor, C#, Advanced

10/18/2019 • 2 minutes to read • [Edit Online](#)

Use the **Advanced** options page to modify the settings for editor formatting, code refactoring, and XML documentation comments for C#. To access this options page, choose **Tools** > **Options**, and then choose **Text Editor** > **C#** > **Advanced**.

NOTE

Not all options may be listed here.

Analysis

- Enable full solution analysis

Enables code analysis on all files in the solution, not just open code files. For more information, see [Full solution analysis](#).

Using Directives

- Place 'System' directives first when sorting usings

When selected, the **Remove and Sort Usings** command in the right-click menu sorts the `using` directives and places the 'System' namespaces at the top of the list.

Before sorting:

```
using AutoMapper;
using FluentValidation;
using System.Collections.Generic;
using System.Linq;
using Newtonsoft.Json;
using System;
```

After sorting:

```
using System;
using System.Collections.Generic;
using System.Linq;
using AutoMapper;
using FluentValidation;
using Newtonsoft.Json;
```

- Separate using directive groups

When selected, the **Remove and Sort Usings** command in the right-click menu separates `using` directives by inserting an empty line between groups of directives that have the same root namespace.

Before sorting:

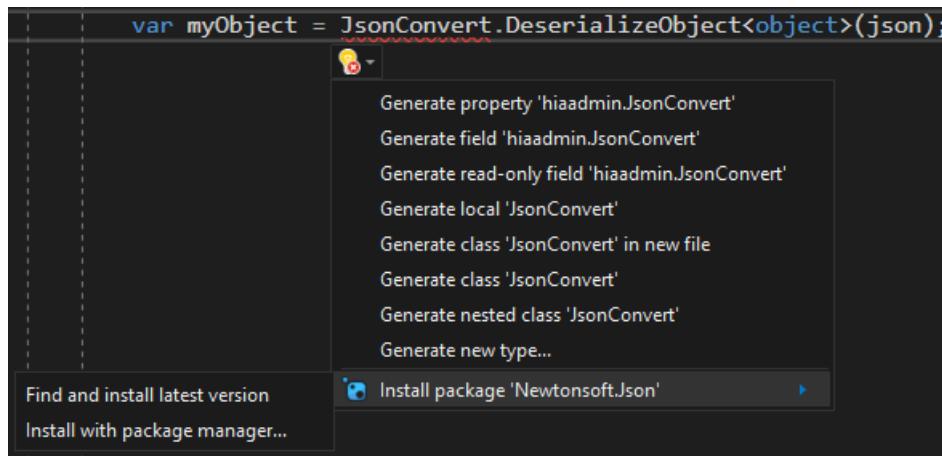
```
using AutoMapper;
using FluentValidation;
using System.Collections.Generic;
using System.Linq;
using Newtonsoft.Json;
using System;
```

After sorting:

```
using AutoMapper;
using FluentValidation;
using Newtonsoft.Json;
using System;
using System.Collections.Generic;
using System.Linq;
```

- Suggest usings for types in reference assemblies
- Suggest usings for types in NuGet packages

When these options are selected, a [Quick Action](#) is available to install a NuGet package and add a `using` directive for unreferenced types.



Highlighting

- Highlight references to symbol under cursor

When the cursor is positioned inside a symbol, or when you click a symbol, all the instances of that symbol in the code file are highlighted.

Outlining

- Enter outlining mode when files open

When selected, automatically outlines the code file, which creates collapsible blocks of code. The first time a file is opened, #regions blocks and inactive code blocks collapse.

- Show procedure line separators

The text editor indicates visual scope of procedures. A line is drawn in the .cs source files of your project at locations listed in the following table:

LOCATION IN .CS SOURCE FILE	EXAMPLE OF LINE LOCATION
After the close of a block declaration construct	<ul style="list-style-type: none"> - At the end of a class, structure, module, interface, or enum - After a property, function, or sub - Not between the get and set clauses in a property
After a set of single line constructs	<ul style="list-style-type: none"> - After the import statements, before a type definition in a class file - After variables declared in a class, before any procedures
After single line declarations (non-block level declarations)	<ul style="list-style-type: none"> - Following import statements, inherits statements, variable declarations, event declarations, delegate declarations, and DLL declare statements

Block Structure Guides

Select these check boxes to display dotted vertical lines between the curly brackets ({}) in your code. You can then easily see individual blocks of code for your declaration level and code level constructs.

Editor Help

- Generate XML documentation comments for `///`

When selected, inserts the XML elements for XML documentation comments after you type the `///` comment introduction. For more information about XML documentation, see [XML Documentation Comments \(C# Programming Guide\)](#).

See also

- [How to: Insert XML comments for documentation generation](#)
- [XML Documentation Comments \(C# Programming Guide\)](#)
- [Document your code with XML comments \(C# Guide\)](#)
- [Set language-specific editor options](#)
- [C# IntelliSense](#)

Options dialog box: Text Editor > C# > Code Style > Formatting

10/18/2019 • 4 minutes to read • [Edit Online](#)

Use the **Formatting** options page and its subpages (**Indentation**, **New Lines**, **Spacing**, and **Wrapping**) to set options for formatting code in the code editor.

To access this options page, choose **Tools > Options** from the menu bar. In the **Options** dialog box, choose **Text Editor > C# > Code Style > Formatting**.

TIP

The **Indentation**, **New Lines**, **Spacing**, and **Wrapping** subpages each display a preview window at the bottom that shows the effect of each option. To use the preview window, select a formatting option. The preview window shows an example of the selected option. When you change a setting by selecting a radio button or check box, the preview window updates to show the effect of the new setting.

Formatting (General) page

General settings

These settings affect *when* the code editor applies formatting options to code.

LABEL	DESCRIPTION
Automatically format when typing	When deselected, the format statement on ; and format block on } options are disabled.
Automatically format statement on ;	When selected, formats statements at completion according to the formatting options selected for the editor.
Automatically format block on }	When selected, formats code blocks according to the formatting options selected for the editor as soon as you complete the code block.
Automatically format on return	When selected, formats text when Enter is pressed, to fit the formatting options selected for the editor.
Automatically format on paste	When selected, formats text that is pasted into the editor to fit the formatting options selected for the editor.

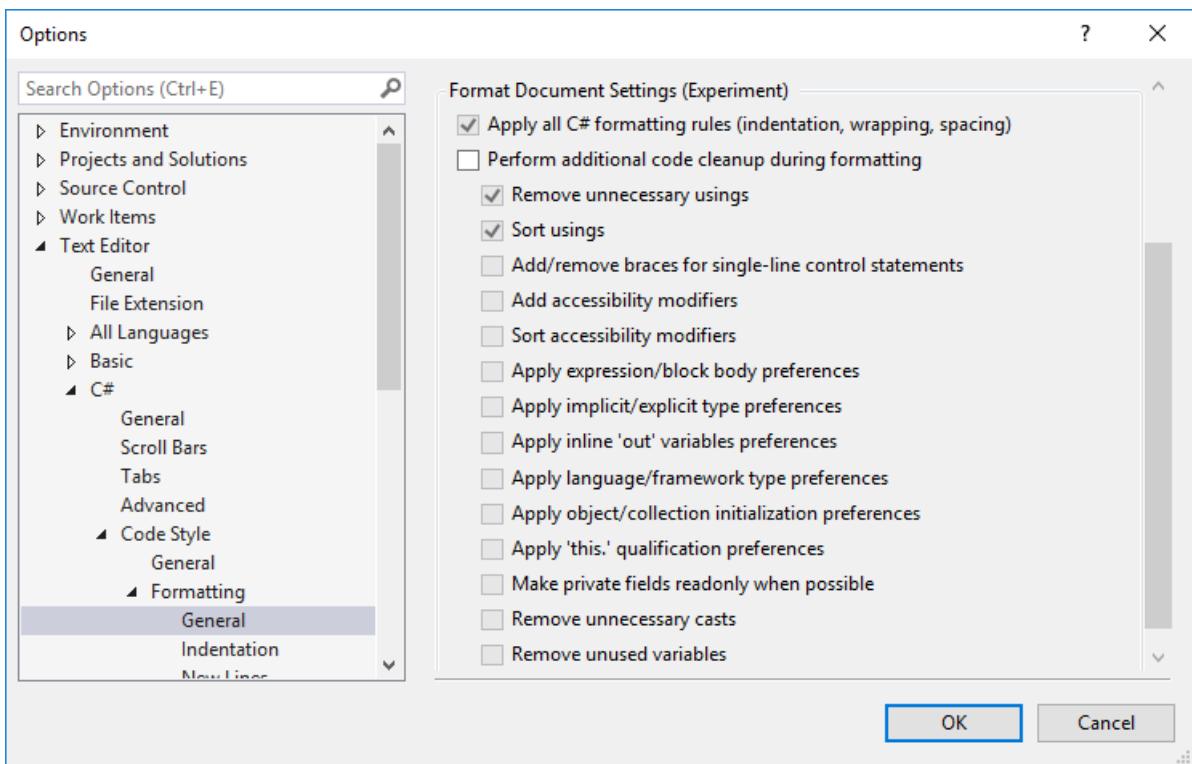
If you previously applied code style settings for C# files using the **Format Document** command in Visual Studio 2017, that functionality is now available as [Code Cleanup](#).

Format Document settings

These settings configure the **Format Document** command to perform additional code cleanup on a file. For more information about how these settings are applied, see [Format Document command](#).

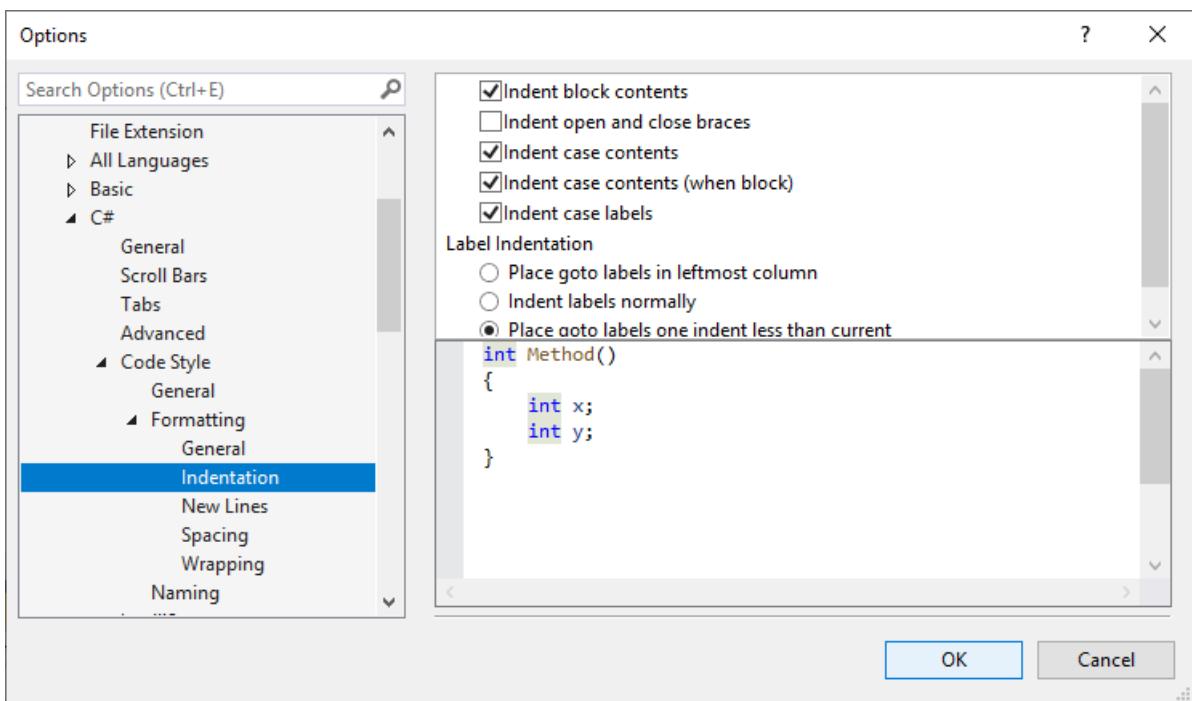
LABEL	DESCRIPTION	CORRESPONDING EDITORCONFIG AND TOOLS > OPTIONS RULES
Apply all C# formatting rules (indentation, wrapping, spacing)	The Format Document command always fixes formatting issues. This setting can't be changed.	Core EditorConfig options .NET EditorConfig formatting options Tools > Options > Text Editor > C# > Formatting > [Indentation or New Lines or Spacing or Wrapping]
Perform addition code cleanup during formatting	When selected, applies fixes for the rules specified below on the Edit.FormatDocument command.	N/A
Remove unnecessary usings	When selected, removes unnecessary <code>using</code> directives when Edit.FormatDocument is triggered.	N/A
Sort usings	When selected, sorts <code>using</code> directives when Edit.FormatDocument is triggered.	dotnet_sort_system_directives_first Tools > Options > Text Editor > C# > Advanced > Place 'System' directives first when sorting usings
Add/remove braces for single-line control statements	When selected, adds or removes braces from single-line control statements when Edit.FormatDocument is triggered.	csharp_prefer_braces Tools > Options > Text Editor > C# > Code Style > Code block preferences > Prefer braces
Add accessibility modifiers	When selected, adds missing accessibility modifiers when Edit.FormatDocument is triggered.	dotnet_style_require_accessibility_modifiers
Sort accessibility modifiers	When selected, sorts accessibility modifiers when Edit.FormatDocument is triggered.	csharp_preferred_modifier_order visual_basic_preferred_modifier_order
Apply expression/block body preferences	When selected, converts expression-bodied members to block bodies, or vice versa, when Edit.FormatDocument is triggered.	Expression-bodied member EditorConfig options Tools > Options > Text Editor > C# > Code Style > Expression preferences > Use expression body for methods, constructors, etc.
Apply implicit/explicit type preferences	When selected, converts <code>var</code> to the explicit type, or vice versa, when Edit.FormatDocument is triggered.	Explicit type EditorConfig options Tools > Options > Text Editor > C# > Code Style > 'var' preferences
Apply inline 'out' variables preferences	When selected, inlines <code>out</code> variables where possible when Edit.FormatDocument is triggered.	csharp_style_inlined_variable_declaratio n Tools > Options > Text Editor > C# > Code Style > Variable preferences > Prefer inlined variable declaration

LABEL	DESCRIPTION	CORRESPONDING EDITORCONFIG AND TOOLS > OPTIONS RULES
Apply language/framework type preferences	When selected, converts language types to framework types, or vice versa, when Edit.FormatDocument is triggered.	dotnet_style_predefined_type_for_locals_parameters_members dotnet_style_predefined_type_for_member_access Tools > Options > Text Editor > C# > Code Style > predefined type preferences
Apply object/collection initialization preferences	When selected, uses object and collection initializers where possible when Edit.FormatDocument is triggered.	dotnet_style_object_initializer dotnet_style_collection_initializer Tools > Options > Text Editor > C# > Code Style > Expression preferences > Prefer object initializer or Prefer collection initializer
Apply 'this.' qualification preferences	When selected, applies <code>this.</code> preferences when Edit.FormatDocument is triggered.	this. qualification EditorConfig options Tools > Options > Text Editor > C# > Code Style > 'this.' preferences
Make private fields readonly when possible	When selected, makes private fields <code>readonly</code> where possible when Edit.FormatDocument is triggered.	dotnet_style_READONLY_FIELD Tools > Options > Text Editor > C# > Code Style > Field preferences > Prefer readonly
Remove unnecessary casts	When selected, removes unnecessary casts where possible when Edit.FormatDocument is triggered.	N/A
Remove unused variables	When selected, removes variables that are unused when Edit.FormatDocument is triggered.	N/A



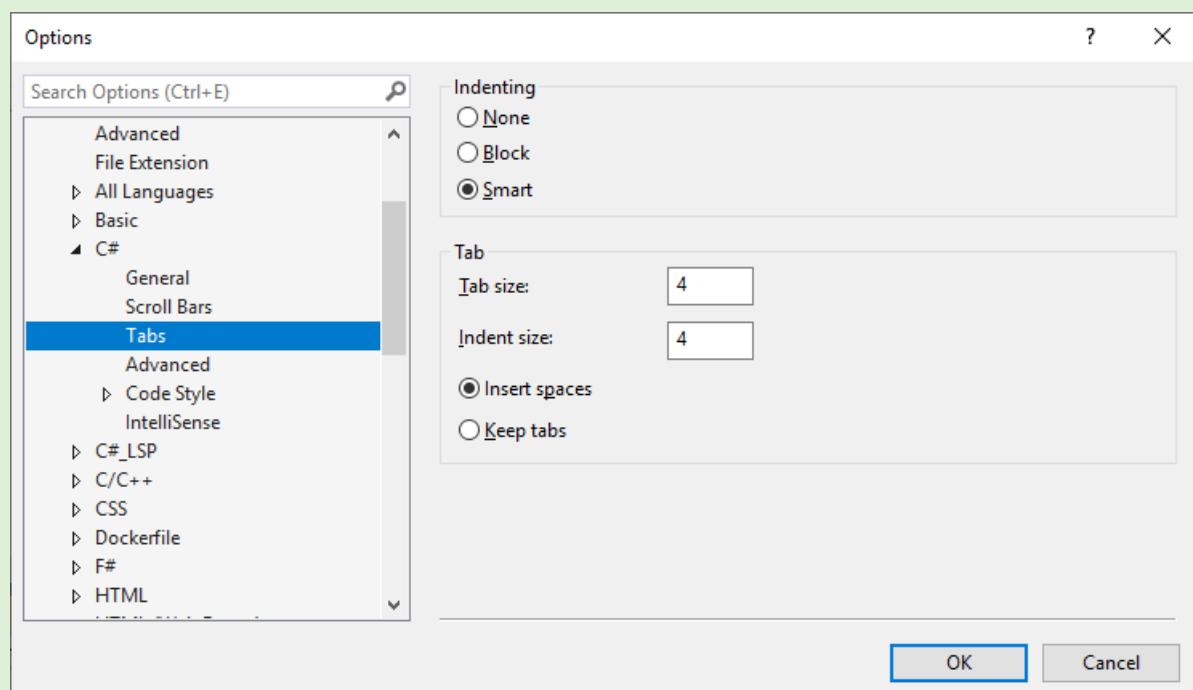
Indentation page

The indentation options on this page apply when code is formatted automatically. One example of when code is automatically formatted is when you paste code into the file while **Automatically format on paste** is selected. (The **Automatically format on paste** option is under **Formatting > General**.)



TIP

There are also indentation options on the **Text Editor > C# > Tabs** options page. Those options only determine where the code editor places the cursor when you press **Enter** at the end of a line.



See also

- [General, Environment, Options dialog box](#)

Options, Text Editor, C#, IntelliSense

10/18/2019 • 2 minutes to read • [Edit Online](#)

Use the **IntelliSense** options page to modify settings that affect the behavior of IntelliSense for C#. To access this options page, choose **Tools > Options**, and then choose **Text Editor > C# > IntelliSense**.

The **IntelliSense** options page contains the following options:

Completion Lists

- Show completion list after a character is typed*

When this option is selected, IntelliSense automatically displays the completion list when you begin typing. When this option is not selected, IntelliSense completion is still available from the **IntelliSense** menu or by pressing **Ctrl+Space**.

- Show completion list after a character is deleted
- Highlight matching portions of completion list items
- Show completion item filters

Snippets behavior

- Never include snippets

When this option is selected, IntelliSense never adds aliases for C# code snippets to the completion list.

- Always include snippets

When this option is selected, IntelliSense adds aliases for C# code snippets to the completion list. In the case where the code snippet alias is the same as a keyword, for example, `class`, the keyword is replaced by the shortcut. For more information, see [C# Code Snippets](#).

- Include snippets when ?-Tab is typed after an identifier

When this option is selected, IntelliSense adds aliases for C# code snippets to the completion list when **?+Tab** is pressed after an identifier

Enter key behavior

- Never add new line on enter

Specifies that a new line is never added automatically after selecting an item in the completion list and pressing **Enter**.

- Only add new line on enter after end of fully typed word

Specifies that if you type all the characters for an entry in the completion list and then press **Enter**, a new line is added automatically and the cursor moves to the new line.

For example, if you type `else` and then press **Enter**, the following appears in the editor:

```
else
```

```
| (cursor location)
```

However, if you type only `el` and then press **Enter**, the following appears in the editor:

`else| (cursor location)`

- Always add new line on enter

Specifies that if you type *any* of the characters for an entry in the completion list and then press **Enter**, a new line is added automatically and the cursor moves to the new line.

Show name suggestions

Performs automatic object name completion for the members that you have recently selected.

See also

- [General, Environment, Options Dialog Box](#)
- [Using IntelliSense](#)

Options, Text Editor, C/C++, Advanced

10/21/2019 • 8 minutes to read • [Edit Online](#)

By changing these options, you can change the behavior related to IntelliSense and the browsing database when you're programming in C or C++.

To access this page, in the **Options** dialog box, in the left pane, expand **Text Editor**, expand **C/C++**, and then choose **Advanced**.

NOTE

Your computer might show different names or locations for some of the Visual Studio user interface elements in the following instructions. The Visual Studio edition that you have and the settings that you use determine these elements. See [Personalize the Visual Studio IDE](#).

Browsing/Navigation

You should never choose these options except in the rare case where a solution is so large that the database activity consumes an unacceptable amount of system resources.

Disable Database

All use of the code browsing database (SDF), all other Browsing/Navigation options, and all IntelliSense features except for #include Auto Complete are disabled.

Disable Database Updates

The database will be opened read-only, and no updates will be performed as files are edited. Most features will still work. However, as edits are made, the data will become stale, and you'll get incorrect results.

Disable Database Auto Updates

The code browsing database won't be automatically updated when source files are modified. However, if you open **Solution Explorer**, open the shortcut menu for the project, and then choose **Rescan Solution**, all out-of-date files will be checked, and the database will be updated.

Disable Implicit Files

The code browsing database doesn't collect data for files that aren't specified in a project. A project contains source files and header files that are explicitly specified. Implicit files are included by explicit files (for example, afxwin.h, windows.h, and atlbase.h). Normally, the system finds these files and also indexes them for various browsing features (including Navigate To). If you choose this option, those files aren't indexed, and some features aren't available for them. If you choose this option, "Disable Implicit Cleanup" and "Disable External Dependencies" are also implicitly chosen.

Disable Implicit Cleanup

The code browsing database doesn't clean up implicit files that are no longer referenced. This option prevents implicit files from being removed from the database when they're no longer used. For example, if you add an `#include` directive that references mapi.h to one of your source files, mapi.h will be found and indexed. If you then remove the `#include` and the file isn't referenced elsewhere, information about it will eventually be removed unless you choose this option. (See the **Rescan Solution Interval** option.) This option is ignored when you explicitly rescan the solution.

Disable External Dependencies Folders

The External Dependencies folder for each project isn't created or updated. In **Solution Explorer**, each project contains an External Dependencies folder, which contains all implicit files for that project. If you choose this option, that folder doesn't appear.

Recreate Database

Recreate the code browsing database from nothing the next time that the solution loads. If you choose this option, the SDF database file is deleted the next time you load the solution, thus causing the database to be recreated and all files indexed.

Rescan Solution Interval

A 'Rescan Solution Now' job is scheduled for the interval that you specify. You must specify between 0 and 5000 minutes. The default value is 60 minutes. While the solution is rescanned, file timestamps are checked to determine whether a file was changed outside of the IDE. (Changes that are made in the IDE are automatically tracked, and files are updated.) Implicitly included files are checked to determine whether they're all still referenced.

Diagnostic Logging

These options are provided in case Microsoft asks you to collect advanced information to diagnose an issue. The logging information isn't useful for users, and we recommend that you leave it disabled.

Enable Logging

Enables diagnostic logging to the output window.

Logging Level

Set the log verbosity, from 0 to 5.

Logging Filter

Filters displayed event types by using a bitmask.

Set by using a sum of any of the following options:

- 0 - None
- 1 - General
- 2 - Idle
- 4 - WorkItem
- 8 - IntelliSense
- 16 - ACPerf
- 32 - ClassView

Fallback Location

The fallback location is where the SDF and IntelliSense support files (for example, iPCH) are put when the primary location (same directory as solution) isn't used. This situation could occur the user doesn't have the permissions to write to the solution directory or the solution directory is on a slow device. The default fallback location is in the user's temp directory.

Always Use Fallback Location

Indicates that the code browsing database and IntelliSense files should always be stored in a folder that you

specify as your "Fallback Location", not next to the .sln file. The IDE will never try to put the SDF or iPCH files next to the solution directory and will always use the fallback location.

Do Not Warn If Fallback Location Used

You aren't informed or prompted if a 'Fallback Location' is used. Normally, the IDE will tell you if it had to use the fallback location. This option turns off that warning.

Fallback Location

This value is used as a secondary location to store the code browsing database or IntelliSense files. By default, your temporary directory is your fallback location. The IDE will create a subdirectory under the specified path (or the temp directory) that includes the name of the solution along with a hash of the full path to the solution, which avoids issues with solution names being identical.

IntelliSense

Auto Quick Info

Enables QuickInfo tooltips when you move the pointer over text.

Disable IntelliSense

Disables all IntelliSense features. The IDE does not create VCPkgSrv.exe processes to service IntelliSense requests, and no IntelliSense features will work (QuickInfo, Member List, Auto Complete, Param Help). Semantic colorization and reference highlighting are also disabled. This option doesn't disable browsing features that rely solely on the database (including Navigation Bar, ClassView, and Property window).

Disable Auto Updating

IntelliSense updating is delayed until an actual request for IntelliSense is made. This delay can result in a longer execution time of the first IntelliSense operation on a file, but it may be helpful to set this option on very slow or resource-constrained machines. If you choose this option, you also implicitly choose the "Disable Error Reporting" and "Disable Squiggles" options.

Disable Error Reporting

Disables reporting of IntelliSense errors through squiggles and the Error List window. Also disables the background parsing that's associated with error reporting. If you choose this option, you also implicitly choose the "Disable Squiggles" option.

Disable Squiggles

Disables IntelliSense error squiggles. The red "squiggles" don't show in the editor window, but the error will still appear in the Error List window.

Auto Tune Max Cached Translation Units

The maximum number of translation units that will be kept active at any one time for IntelliSense requests. You must specify a value between 2 and 15. This number directly relates to the maximum number of VCPkgSrv.exe processes that will run (for a given instance of Visual Studio). The default value is 2, but if you have available memory, you can increase this value and possibly achieve slightly better performance on IntelliSense.

For more information about translation units, see [Phases of Translation](#).

Disable #include Auto Complete

Disables auto-completion of `#include` statements.

Use Forward Slash in #include Auto Complete

Triggers auto-completion of `#include` statements when "/" is used. The default delimiter is backslash ". The compiler can accept either, so use this option to specify what your code base uses.

Disable Aggressive Member List

The member list doesn't appear while you type the name of a type or variable. The list appears only after you type one of the commit characters, as defined in the **Member List Commit Characters** option.

Disable Member List Keywords

Language keywords such as `void`, `class`, `switch` don't appear in member list suggestions.

Disable Member List Code Snippets

Code snippets don't appear in member list suggestions.

Member List Filter Mode

Sets the type of matching algorithm. **Fuzzy** finds the most possible matches because it uses an algorithm that's similar to a spell-checker to find matches that are similar but not identical. **Smart filtering** matches substrings even if they're not at the start of a word. **Prefix** only matches on identical substrings that start at the beginning of the word.

Disable Semantic Colorization

Turns off all code colorization except for language keywords, strings, and comments.

Member List Commit Characters

Specifies the characters that cause the currently highlighted Member List suggestion to be committed. You can add or remove characters from this list.

Smart Member List Commit

Adds a line when you choose the Enter key at the end of a fully typed word.

Enable Member List Dot-To-Arrow

Replaces '.' with '->' when applicable for Member List.

References

Disable Resolving

For performance reasons, 'Find All References' displays raw textual search results by default instead of using IntelliSense to verify each candidate. You can clear this check box for more accurate results on all find operations. To filter on a per-search basis, open the shortcut menu for the result list, and then choose "Resolve Results."

Hide Unconfirmed

Hide unconfirmed items in the 'Find All References' results. If you unset the "Disable Resolving" option, you can use this option to hide unconfirmed items in the results.

Disable Reference Highlighting

By default, when you select some text, all instances of the same text are automatically highlighted in the current document. You can disable this feature by setting **Disable Reference Highlighting** to **True**.

Text Editor

Enable Surround with Braces

If enabled, you can surround selected text with curly braces by typing '{' into the text editor.

Enable Surround with Parentheses

If enabled, you can surround selected text with parentheses by typing '(' into the text editor.

See also

- [Setting Language-Specific Editor Options](#)

Options, Text Editor, C/C++, Experimental

7/24/2019 • 2 minutes to read • [Edit Online](#)

By changing these options, you can change the behavior related to IntelliSense and the browsing database when you're programming in C or C++. These features are truly experimental and may be modified or removed from Visual Studio in a future release.

This article describes the options in Visual Studio 2017. For Visual Studio 2015, select **2015** in the selector above the table of contents.

To access this property page, press **Ctrl+Q** to activate the search box and then type **experimental**. Search finds the page after the first few letters. You can also get to it by choosing **Tools > Options** and expanding **Text Editor**, then **C/C++**, and then choosing **Experimental**.

These features are available in a Visual Studio installation.

NOTE

Your computer might show different names or locations for some of the Visual Studio user interface elements in the following instructions. The Visual Studio edition that you have and the settings that you use determine these elements. See [Personalize the Visual Studio IDE](#).

Enable Predictive IntelliSense

Predictive IntelliSense limits the number of results displayed in the IntelliSense dropdown list so that you see only results that are relevant in the context. For example, if you type `int x =` and invoke the IntelliSense dropdown, you will see only integers or functions that return integers. Predictive IntelliSense is turned off by default.

Enable faster project load

As of Visual Studio 2017 version 15.3, this feature is called **Enable Project Caching** and has moved to the [VC++ Project Settings](#) property page.

This option enables Visual Studio to cache project data so that when you open the project the next time, it can load that cached data rather than re-computing it from the project files. Using cached data can speed up the project load time significantly.

Additional features in the Visual Studio Marketplace

You can browse additional text editor features in the [Visual Studio Marketplace](#). An example is [C++ Quick Fixes](#), which supports the following:

- **Add missing #include** - Suggests relevant #include's for unknown symbols in your code
- **Add using namespace/Fully qualify symbol** - Like the previous item, but for namespaces
- **Add missing semicolon**
- **Online help** - Search online help for your error messages
- And more...

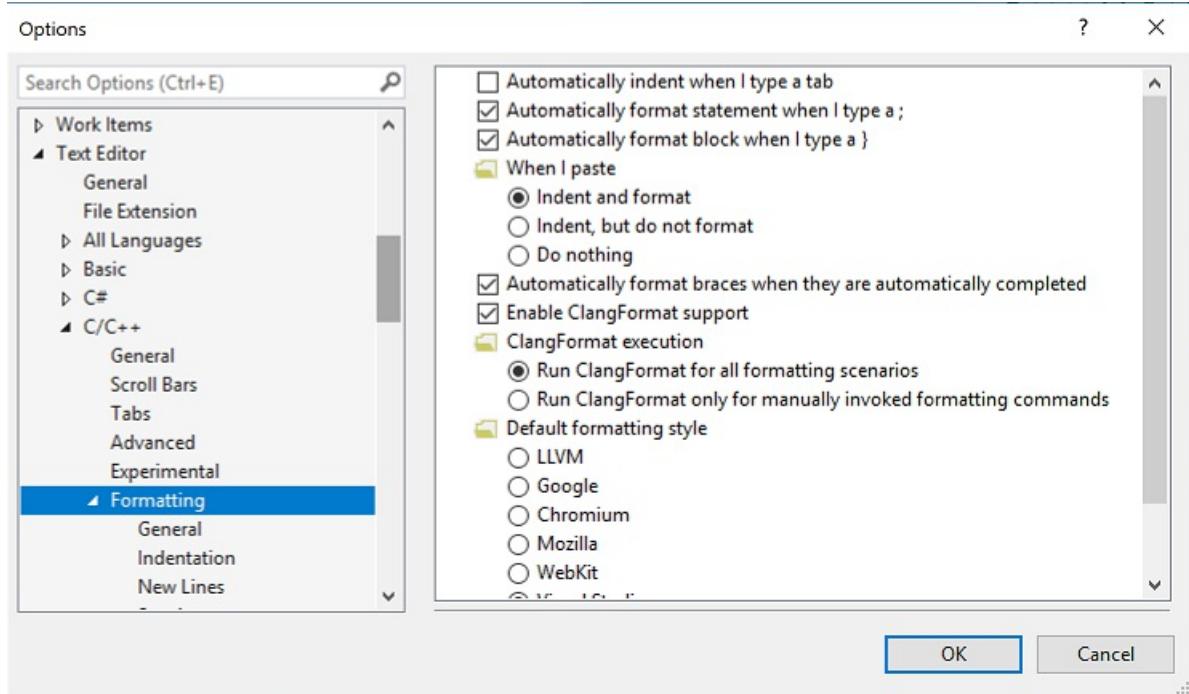
See also

- [Setting Language-Specific Editor Options](#)
- [Refactoring in C++ \(VC Blog\)](#)

Options, Text Editor, C/C++, Formatting

7/24/2019 • 2 minutes to read • [Edit Online](#)

Use these property pages to change the default behavior of the code editor when you are programming in C or C++.



To access this page, in the **Options** dialog box, in the left pane, expand **Text Editor**, expand **C/C++**, and then click **Formatting**.

NOTE

Your computer might show different names or locations for some of the Visual Studio user interface elements in the following instructions. The Visual Studio edition that you have and the settings that you use determine these elements. For more information, see [Personalize the Visual Studio IDE](#).

General Page

This page has options for formatting statements and blocks as you type them.

Visual Studio 2017 version 15.7 and later:

The page also has options for configuring support for [ClangFormat](#) version 5.0. ClangFormat is a utility that makes it easy to style and format your code based on a set of rules that can be configured in a .clang-format or _clang-format file.

Configuring ClangFormat options

Visual Studio 2017 version 15.7 and later:

ClangFormat support is enabled by default. You can choose which of these common formating conventions to apply to all your projects: LLVM, Google, Chromium, Mozilla, or WebKit. You can also create a custom format definition .clang-format or _clang-format file. If such a file is present in a project folder, Visual Studio uses it to format all source code files in that folder and its subfolders.

By default, Visual Studio runs clangformat.exe in the background applies formatting as you type. You can also specify to run it only for manually invoked formatting commands **Format Document (Ctrl+K, Ctrl+D)** or **Format Selection (Ctrl + K, Ctrl + F)**.

Indentation, New Lines, Spacing Wrapping pages

These pages enable various formatting customizations but are ignored if ClangFormat is enabled.

See also

- [General, Environment, Options Dialog Box](#)
- [Using IntelliSense](#)

Options, Text Editor, C/C++, View

10/25/2019 • 2 minutes to read • [Edit Online](#)

Use these property pages to change the default behavior of the code editor when you are programming in C or C++.

To access this property page, choose **Tools > Options** and expand **Text Editor**, then **C/C++**, and then choose **View**.

Code Squiggles

You can enable or disable the following settings to manage the way in which text editor handles code squiggles for C and C++:

- **Macros in Skipped Browsing Regions** - Defines how to highlight macros that are inside skipped regions by the browsing database, such as macros whose definitions include braces.
- **Macros Convertible to constexpr** - Defines how to highlight macro definitions that can be converted to `constexpr` definitions.

Inactive Code

- **Show Inactive Blocks** - Preprocessor inactive blocks are colorized differently.
- **Disable Inactive Code Opacity** - A solid color, instead of opacity, is used for inactive code blocks.
- **Inactive Code Opacity Percent** - The percentage of opacity for inactive code blocks.

Miscellaneous

- **Enumerate Comment Tasks** - Scan open source files for VS tokens and report them in the Task List window.
- **Highlight Matching Tokens** - Highlight enclosing braces or syntax that match where the cursor is positioned.

Outlining

- **Enable Outlining** - Enter outlining mode when a file opens.
- **OutlinePragmaRegions** - Automatically outline `#pragma` region blocks.
- **OutlineStatementBlocks** - Automatically outline statement blocks.

See also

- [Setting Language-Specific Editor Options](#)
- [Refactoring in C++ \(VC Blog\)](#)

Options, Text Editor, F#, Advanced

10/18/2019 • 2 minutes to read • [Edit Online](#)

Use the **Advanced** options page to modify some text editor settings for F#. To access this options page, choose **Tools > Options**, and then choose **Text Editor > F# > Advanced**.

Block Structure Guides

When selected, vertical lines appear in the editor that line up with structured code blocks, which lets you easily identify the individual blocks of code.

Outlining

Select this check box to automatically outline the code file, which creates collapsible blocks of code.

See also

- [General, Environment, Options Dialog Box](#)
- [Options, Text Editor, All Languages, Tabs](#)

Options: Text Editor > F# > Code Fixes

10/18/2019 • 2 minutes to read • [Edit Online](#)

Use the Code Fixes options page to specify the settings that can help identify code errors and offer solutions. To access this options page, choose **Tools** > **Options**, and then choose **Text Editor** > **F#** > **Code Fixes**.

Code Fixes

- **Simplify names (remove unnecessary qualifiers)**

If this check box is selected, fully qualified names are simplified when the qualifications aren't necessary, such as for a member of a frequently used namespace.

- **Always place open statements at the top level**

If this check box is selected and you type an `open` statement in the code, it's put at the top level.

- **Remove unused open statements**

If this check box is selected, documents are analyzed for unused `open` statements, and a **Quick Action** light bulb appears with an action to remove all unused `open` statements.

- **Analyze and suggest fixes for unused values**

If this check box is selected, the tool recognizes a value that isn't being used in the code. Then, if you hover over the unused value, it recommends ways in which you can use the value.

See also

- [General, Environment, Options Dialog Box](#)
- [Find code changes and other history with CodeLens](#)

Options, Text Editor, F#, CodeLens

10/18/2019 • 2 minutes to read • [Edit Online](#)

Use the **CodeLens** options page to modify the text editor CodeLens settings for F#. To access this options page, choose **Tools > Options**, and then choose **Text Editor > F# > CodeLens**.

CodeLens

- **Enable CodeLens (Experimental)**

When this option is selected, interactive annotations appear in the code that let you easily find references and changes to your code, linked bugs, work items, code reviews, and unit tests.

- **Use colors in annotations**

Select this check box to display the CodeLens annotations in color to easily differentiate the items CodeLens reports.

- **Show annotations to the right instead of above the line**

By default, CodeLens annotations appear over a code line. Select this check box to have the annotations appear to the right of the code instead of above it.

- **Annotation prefix**

You can change the default prefix that is used for the annotations.

See also

- [General, Environment, Options Dialog Box](#)
- [Find code changes and other history with CodeLens](#)

Options, Text Editor, F#, IntelliSense

10/18/2019 • 2 minutes to read • [Edit Online](#)

Use the **IntelliSense** options page to modify some text editor settings for F#. To access this options page, choose **Tools > Options**, and then choose **Text Editor > F# > IntelliSense**.

Completion Lists

- **Show completion list after a character is typed**

When this option is selected, IntelliSense automatically displays the completion list when you begin typing. If you don't select this option, IntelliSense completion is still available from the IntelliSense menu or by pressing **Ctrl + Space**.

- **Show completion list after a character is deleted**

When this option is selected, IntelliSense automatically displays a relevant completion list when you begin deleting characters in your code.

- **Show symbols in unopened namespaces**

When this option is selected, IntelliSense automatically displays the completion list when you begin typing and includes items from namespaces that you haven't opened.

See also

- [General, Environment, Options Dialog Box](#)
- [Using IntelliSense](#)

Options, Text Editor, HTML (Web Forms), Formatting

10/18/2019 • 2 minutes to read • [Edit Online](#)

Use the **Formatting** options page to set HTML project options for formatting code in the Code Editor. To access this page, on the menu bar, choose **Tools > Options**, and then expand **Text Editor > HTML (Web Forms) > Formatting**.

Capitalization

When these options are selected, the Source view and XML editors apply a default case format to the names of elements and attributes when the elements are first created and during automatic formatting. The **Apply Automatic Formatting** settings determine the time at which automatic reformatting occurs.

WARNING

XML is case-sensitive. Setting a default case can impact XML parsers.

UIElement list

Server tag, Server attributes

These options specify how the markup for Web server controls is capitalized.

OPTION	RESULT
As entered	Element case is exactly as you enter it.
Uppercase	Element names are reformatted to uppercase.
Lowercase	Element names are reformatted to lowercase.
Assembly definition	Element case is determined by how the element is defined in the corresponding type class.

Client tag, Client attributes

These options specify whether automatic formatting changes the names of HTML attributes and properties to uppercase or lowercase, or keeps them as entered.

OPTION	RESULT
As entered	Attribute case is exactly as you enter it.
Uppercase	Attribute names are reformatted to uppercase.
Lowercase	Attribute names are reformatted to lowercase.

Automatic formatting options

These options cause the Source view editor to add or remove physical line breaks during automatic formatting. You can also specify whether the editor adds quotes around attributes.

NOTE

These settings do not alter white space within XML markup.

UIElement list

- **Insert attribute value quotes when typing**

When this option is selected, the editor automatically puts quotation marks around attributes as you're typing (for example: ID="Select1"). Clear this option if you prefer to insert quotation marks into your markup manually.

NOTE

Whether or not this option is selected, all existing quotation marks in your markup are retained; quotation marks are never removed.

- **Insert attribute value quotes when formatting**

When this option is selected, automatic formatting adds quotation marks around attribute values (for example: ID="Select1").

NOTE

Whether or not this option is selected, all existing quotation marks in your markup are retained.

- **Auto insert close tag**

When this option is selected, the editor automatically creates a closing tag (for example,) when you close the opening tag.

Tag wrapping

These options determine whether the editor breaks tags up into lines if they go beyond a certain length.

UIElement list

- **Wrap tags when exceeding specified length**

When selected, the editor breaks tags across lines if the tag goes beyond the length you specify in the **Length** text box. This action occurs only when formatting the tag, not when you're typing a new tag.

NOTE

The value you specify is used as a minimum value. The editor does not break up individual attributes.

- **Length**

Specifies the number of characters to display in a line before wrapping. This input box is disabled unless the **Wrap tags when exceeding specified length** box is checked.

- **Tag Specific Options**

Displays the **Tag Specific Options** dialog box, which lets you set formatting options for individual tags or groups of tags.

See also

- [General, Environment, Options Dialog Box](#)

Options, Text Editor, HTML (Web Forms), Miscellaneous

10/18/2019 • 2 minutes to read • [Edit Online](#)

Use the **Miscellaneous** options page to set preferences for how the HTML editor handles elements pasted into the page. To access this page, on the menu bar, choose **Tools** > **Options**, and then expand **Text Editor** > **HTML (Web Forms)** > **Miscellaneous**.

Miscellaneous HTML options

- **Format HTML on paste**

When this check box is selected, the editor reformats elements pasted into the page, using currently defined formatting rules. If this check box isn't selected, the editor pastes the elements, using the exact formatting of the original element.

- **Require '<' to trigger tag completion window**

When this check box is selected, the tag completion window appears when you type the less than character ("<").

See also

- [General, Environment, Options Dialog Box](#)

Options, Text Editor, HTML (Web Forms), Validation

10/18/2019 • 2 minutes to read • [Edit Online](#)

Use the **Validation** options page to set preferences for how the HTML editor checks the syntax of HTML markup in your document. To access this page, on the menu bar, choose **Tools > Options**, and then expand **Text Editor > HTML (Web Forms) > Validation**.

Validation

- **Use doctype for validation schema detection**

A schema determines which elements, attributes, and capitalization are valid in that schema. It also determines the tags and attributes that are available in IntelliSense.

Select this option if you want Visual Studio to use the content of the page's `<!DOCTYPE>` declaration and `html` element to determine the schema. For example, if you select this option and the page has the declaration `<!DOCTYPE html>`, Visual Studio uses the HTML5 schema. However, if the `html` tag has an `xmlns` attribute, such as `<html xmlns="http://www.w3.org/1999/xhtml">`, Visual Studio uses the XHTML5 schema.

- **Target when no doctype found**

Choose the schema to validate against when there's no `<!DOCTYPE>` declaration in the page.

- **Show errors**

Select the check box to enable validation. If the check box isn't selected, the editor doesn't mark validation errors.

The other check boxes let you fine-tune validation by specifying individual types of errors that you want the editor to mark.

NOTE

Some schemas do not offer options to mark individual types of errors. For example, if you choose **XHTML 1.1** as the target schema, all option check boxes are disabled. In this instance, all types of errors are marked.

See also

- [General, Environment, Options Dialog Box](#)

Options dialog box: Text Editor > JavaScript/TypeScript > Code Validation

7/29/2019 • 2 minutes to read • [Edit Online](#)

Use the **Code Validation** page of the **Options** dialog box to modify settings that affect the way that JavaScript handles errors. You can access the **Code Validation** page by choosing **Tools** > **Options** on the menu bar, and then expanding **Text Editor** > **JavaScript/TypeScript** > **Code Validation**.

NOTE

Your computer might show different names or locations for some of the Visual Studio user interface elements in this article. You may be using a different edition of Visual Studio or different environment settings. For more information, see [Personalize the IDE](#).

The **Code Validation** page contains the following sections:

JavaScript errors

You can use these options to set preferences for how the JavaScript editor validates syntax in your document.

UIElement list

Enable JavaScript errors

When set to **True**, the JavaScript code editor shows errors for JavaScript and JSX files. Errors appear in the **Error List** with a (JS) prefix. Viewing these errors is useful if you're working with code that you didn't write and you don't intend to fix syntax errors.

Show errors as warnings

When set to **True**, JavaScript errors are shown as warnings instead of errors in the **Error List**.

See also

- [JavaScript IntelliSense](#)

Options dialog box: Text Editor > JavaScript > Formatting

7/29/2019 • 2 minutes to read • [Edit Online](#)

Use the **Formatting** page of the **Options** dialog box to set options for formatting code in the Code Editor. To access this page, on the menu bar, choose **Tools** > **Options**, and then expand **Text Editor** > **JavaScript/TypeScript** > **Formatting**.

NOTE

Your computer might show different names or locations for some of the Visual Studio user interface elements in this article. You may be using a different edition of Visual Studio or different environment settings. For more information, see [Personalize the IDE](#).

Automatic Formatting

These options determine when formatting occurs in **Source** view.

UIElement List

OPTION	DESCRIPTION
Format completed line on Enter	When this option is selected, the Code Editor automatically formats the line when you choose the Enter key.
Format completed statement on ;	When this option is selected, the Code Editor automatically formats the line when you choose the semicolon key.
Format opened block on {	When this option is selected, the Code Editor automatically formats the line when you choose the opening brace key.
Format completed block on }	When this option is selected, the Code Editor automatically formats the line when you choose the closing brace key.
Format on paste	When this option is selected, the Code Editor reformats code when you paste it into the editor. The editor uses the currently defined formatting rules. If this option is not selected, the editor uses the original formatting of the pasted-in code.

New Lines

These options determine whether the Code Editor puts an open brace for functions and control blocks on a new line.

UIElement list

OPTION	DESCRIPTION
Place open brace on new line for functions	When this option is selected, the Code Editor moves the open brace associated with a function to a new line.

OPTION	DESCRIPTION
Place open brace on new line for control blocks	When this option is selected, the Code Editor moves the open brace associated with a control block (for example, <code>if</code> and <code>while</code> control blocks) to a new line.

Spacing

These options determine how spaces are inserted in **Source** view.

UIElement list

OPTION	DESCRIPTION
Insert space after comma delimiter	When this option is selected, the Code Editor adds a space after comma delimiters.
Insert space after semicolon in 'for' statements	When this option is selected, the Code Editor adds a space after each semicolon in the first line of a <code>for</code> loop.
Insert space before and after binary operators	When this option is selected, the Code Editor adds a space before and after binary operators (for example, <code>+</code> , <code>-</code> , <code>&&</code> , <code> </code>).
Insert space after keywords in control flow statements	When this option is selected, the Code Editor adds a space after JavaScript keywords in control flow statements.
Insert space after function keyword for anonymous functions	When this option is selected, the Code Editor adds a space after the <code>function</code> keyword for anonymous functions.
Insert space after opening and before closing non-empty parenthesis	When this option is selected, the Code Editor adds a space after the opening parenthesis and before the closing parenthesis if non-empty characters are present within the parentheses.

See also

- [General, Environment, Options Dialog Box](#)

Options dialog box: Text Editor > JavaScript > IntelliSense

7/29/2019 • 2 minutes to read • [Edit Online](#)

Use the **IntelliSense** page of the **Options** dialog box to modify settings that affect the behavior of IntelliSense for JavaScript. You can access the **IntelliSense** page by choosing **Tools** > **Options** on the menu bar, and then expanding **Text Editor** > **JavaScript/TypeScript** > **IntelliSense**.

NOTE

Your computer might show different names or locations for some of the Visual Studio user interface elements in this article. You may be using a different edition of Visual Studio or different environment settings. For more information, see [Personalize the IDE](#).

The **IntelliSense** page contains the following sections:

Statement Completion

You can use these options to change the behavior of IntelliSense statement completion.

UIElement list

Only use Tab or Enter to commit

When you select this check box, the JavaScript code editor appends statements with items selected in the completion list only after you choose the **Tab** or **Enter** key. When you deselect this check box, other characters – such as a period, comma, colon, open parenthesis, and open brace ({}) – can also append statements with the selected items.

References

You can use these options to specify the types of IntelliSense js files that are in scope for different JavaScript project types. The IntelliSense references are typically used to provide IntelliSense support for global objects. You can also use this page to set the loading order for scripts that must be loaded at run time, and to add IntelliSense extension files.

UIElement list

Reference groups

This option specifies the reference group type. Three reference groups are supported:

You can use pre-defined reference groups to specify that particular IntelliSense js files are in scope for different JavaScript projects. Four reference groups are available:

- Implicit (Windows version), for Windows 8.x Store apps using JavaScript. Files included in this group are in scope for every .js file opened in the Code Editor for Windows 8.x Store apps using JavaScript.
- Implicit (Web), for HTML5 projects. Files included in this group are in scope for every .js file opened in the Code Editor for these project types.
- Dedicated worker reference groups, for HTML5 web workers. Files specified in this group are in scope for .js files that have an explicit reference to a dedicated worker reference group.

- Generic, for other JavaScript project types.

Included files

This option specifies the order in which files are loaded into the context of the language service. You can configure the order by using the **Remove**, **Move Up**, and **Move Down** buttons. For IntelliSense to work correctly, a file that is dependent on another must be loaded after the other file.

Caution

If an object is defined unconditionally in two or more implicit references, the last reference in this list will be used to define the object.

Add a reference to the group

This option provides a way to add additional IntelliSense js files by browsing to the appropriate files.

Download remote references (e.g. `http://`) for files in the miscellaneous files project

When this check box is selected, and if you have a JavaScript file opened outside the context of a project, Visual Studio downloads remote JavaScript files referenced in the file for the purpose of providing IntelliSense information. If this option is selected, files are downloaded when you include them as a reference in your JavaScript file.

NOTE

For web projects, remote files referenced in your project are downloaded by default.

See also

- [JavaScript IntelliSense](#)

Options dialog box: Text Editor > JavaScript/TypeScript > Linting

7/29/2019 • 2 minutes to read • [Edit Online](#)

Use the **Linting** page of the **Options** dialog box to set options for analyzing code in the Code Editor. To access this page, on the menu bar, choose **Tools** > **Options**, and then expand **Text Editor** > **JavaScript/TypeScript** > **Linting**.

ESLint Settings

These options let you enable static JavaScript and TypeScript code analysis, and choose which files are analyzed. For more information about ESLint, see [ESLint.org](#).

UIElement list

OPTION	DESCRIPTION
Enable ESLint	When this option is selected, the Code Editor allows for static analysis on the code.
Lint all files included in project, even closed files	When this option is selected, closed files are analyzed, unless diagnostics are only reported for open files.

Global ESLint Config Options

This option lets you copy the location of the global ESLint configuration file. Also, if you previously changed the location, you can reset the file to its default location.

See also

- [General, Environment, Options Dialog Box](#)

Options, Text Editor, JavaScript, Project

7/29/2019 • 2 minutes to read • [Edit Online](#)

Use the **Project** page of the **Options** dialog box to specify JavaScript and TypeScript project options in the Code Editor. To access this page, on the menu bar, choose **Tools > Options**, and then expand **Text Editor > JavaScript/TypeScript > Project**.

Project Analysis Options

These options determine how the editor analyzes projects, reports diagnostics, and suggests improvements. Select or clear the options to specify how the editor handles these situations.

UIElement list

- **Only analyze projects which contain files opened in the editor**
- **Only report diagnostics for files opened in the editor**
- **Suggest possible improvements that are not corrections**

Virtual Projects in Solution Explorer

These options let you choose whether to display Virtual Projects when a Solution is either loaded or not loaded.

Compile on Save

These options determine whether TypeScript files that aren't part of the project are automatically compiled. Select the check box and then choose the type of code generation to use.

UIElement list

- **Use AMD code generation for modules that are not part of a project**
- **Use CommonJS code generation for modules that are not part of a project**
- **Use UMD code generation for modules that are not part of a project**
- **Use System code generation for modules that are not part of a project**
- **Use ES2015 code generation for modules that are not part of a project**

ECMAScript version for files that are not part of a project

These options lets you select the ECMAScript version for files that aren't part of a project. You can choose between **ECMAScript 3**, **ECMAScript 5**, or **ECMAScript 6**.

JSX Emit for TSX files that are not part of a project

These options determine how the editor treats TypeScript files that aren't part of a project.

UIElement list

OPTION	DESCRIPTION
React Framework	When this option is selected, the Code Editor emits a <code>.js</code> file extension.

OPTION	DESCRIPTION
Preserve	When this option is selected, the Code Editor keeps the JSX as part of the output and emits a <code>.jsx</code> file extension.

See also

- [General, Environment, Options Dialog Box](#)

Options, Text Editor, U-SQL, Formatting

10/18/2019 • 2 minutes to read • [Edit Online](#)

Use the **Formatting** options page to set options for formatting code in the code editor. To access this options page, choose **Tools > Options**. In the **Options** dialog box, choose **Text Editor > U-SQL > Formatting**.

General page

General settings

These settings affect *when* the code editor applies formatting options to code.

- **Automatically format completed statement on entering semicolon**

When selected, formats statements when you choose the semicolon key according to the formatting options selected for the editor.

- **Automatically format on paste**

When selected, formats text that is pasted into the editor to fit the formatting options selected for the editor.

Preview windows

The **Indentation**, **New Lines**, and **Spacing** subpages each display a preview window at the bottom. The preview window shows the effect of each option. To use the preview window, select a formatting option. The preview window shows an example of the selected option. When you change a setting by selecting a check box, the preview window updates to show the effect of the new setting.

Indentation remarks

Indentation options on the **Tabs** pages for each language only determine where the code editor places the cursor when you press **Enter** at the end of a line. Indentation options under **Formatting** apply when code is formatted automatically, for example:

- When you paste code into the file while **Automatically format on paste** is selected
- When the block being formatted is typed manually

See also

- [General, Environment, Options dialog box](#)

Options, Text Editor, U-SQL, IntelliSense

10/18/2019 • 2 minutes to read • [Edit Online](#)

Use the **IntelliSense** options page to modify some text editor settings for U-SQL. To access this options page, choose **Tools > Options**, and then choose **Text Editor > U-SQL > IntelliSense**.

IntelliSense Settings

Select a check box to enable **Quick Info** or **Intellisense**. Quick Info displays the complete declaration when you hover the mouse cursor over a variable.

Completion Lists

- **Show completion list after a character is typed**

When this option is selected, IntelliSense automatically displays the completion list when you begin typing.

If you don't select this option, IntelliSense completion is still available from the IntelliSense menu or by pressing **Ctrl + Spacebar**.

- **Place keywords in completion lists**

When this option is selected, IntelliSense includes keywords in the completion list.

- **Place code snippets in completion lists**

When this option is selected, IntelliSense includes code snippets in the completion list.

Selection in Completion List

- **Commit by typing the following characters**

This field shows the characters that cause the currently highlighted completion list suggestion to be committed. You can add or remove characters from this list.

- **Commit by pressing the Spacebar**

When this option is selected, you can commit the highlighted completion list suggestion by pressing the spacebar.

- **Add a new line at the end of fully typed word on Enter**

When selected, a new line is added automatically and the cursor moves to the new line when you type all the characters for a completion list suggestion.

See also

- [General, Environment, Options Dialog Box](#)
- [Using IntelliSense](#)

Options, Text Editor, XAML, Formatting

10/18/2019 • 2 minutes to read • [Edit Online](#)

Use the **Formatting** property page to specify how elements and attributes are formatted in your XAML documents. To open the **Options** dialog box, click the **Tools** menu and then click **Options**. To access the **Formatting** property page, expand the **Text Editor > XAML > Formatting** node.

Auto-Formatting Events

Autoformatting may occur when any of the following events is detected.

- Completion of an end tag or simple tag.
- Completion of a start tag.
- Pasting from the clipboard.
- Formatting keyboard commands.

You can specify which events cause autoformatting.

On completion of end tag or simple tag

Autoformatting occurs when you finish typing an end tag or a simple tag. A simple tag has no attributes, for example `<Button />`.

On completion of start tag

Autoformatting occurs when you finish typing a start tag.

On paste from clipboard

Autoformatting occurs when you paste XAML from the clipboard into XAML view.

Quotation Mark Style

This setting indicates whether attribute values are enclosed in single or double quotation marks. The autoformatter and IntelliSense autocompletion both use this setting.

Once you set this option, only attributes subsequently added either using the designer or manually in the XAML view are affected.

Double quotes ("")

Attribute values are enclosed in double quotes. `<Button Name="button1">Hello</Button>`

Single quotes ('')

Attribute values are enclosed in single quotes. `<Button Name='button1'>Hello</Button>`

Tag Wrapping

You can specify a line length for tag wrapping. When tag wrapping is enabled, any XAML subsequently added by using the designer will be wrapped appropriately.

Wrap tags that exceed specified length

Specifies whether lines are wrapped at the line length specified by **Length**.

Length

The number of characters a line may contain. If necessary, some XAML lines might exceed the specified line length.

Attribute Spacing

Use this setting to control how attributes are arranged in your XAML document

Preserve newlines and spaces between attributes

New lines and spaces between attributes are not affected by autoformatting.

```
<Button Height="23" Name="button1"  
Width="75">Hello</Button>
```

Insert a single space between attributes

Attributes occupy one line, with one space separating adjacent attributes. Tag wrapping settings are applied.

```
<Button Height="23" Name="button1" Width="75">Hello</Button>
```

Position each attribute on a separate line

Each attribute occupies its own line, which is useful when many attributes are present.

```
<Button  
Height="23"  
Name="button1"  
Width="75">Hello</Button>
```

Position first attribute on same line as start tag

When checked, the first attribute appears on the same line as the element's start tag.

```
<Button Height="23"  
Name="button1"  
Width="75">Hello</Button>
```

Element Spacing

Use this setting to control how elements are arranged in your XAML document.

Preserve new lines in content

Empty lines in element content are not removed.

```
<Grid>  
  
<Button Name="button1">Hello</Button>  
  
</Grid>
```

Collapse multiple empty lines in content to a single line

Empty lines in element content are collapsed to a single line.

```
<Grid>  
  
<Button Name="button1">Hello</Button>  
  
</Grid>
```

Remove empty lines in content

All empty lines in element content are removed.

```
<Grid>  
<Button Name="button1">Hello</Button>  
</Grid>
```

See also

- [XAML in WPF](#)

Options, Text Editor, XAML, Miscellaneous

10/31/2019 • 2 minutes to read • [Edit Online](#)

Use the **Miscellaneous** property page to specify how elements and attributes are formatted in your XAML documents. To open the **Options** dialog box, click the **Tools** menu and then click **Options**. To access the **Miscellaneous** property page, expand the **Text Editor > XAML > Miscellaneous** node.

Auto Insert

Use this setting to control when tags and quotes are automatically generated.

Closing tags	Specifies whether an element's closing tag is automatically generated when you close the opening tag with the greater than character (>).
Attribute quotes	Specifies whether enclosing quotes are generated when an attribute value is selected from the statement completion drop-down list.
Closing braces for MarkupExtensions	Specifies whether a markup extension's closing brace {} is automatically generated when you type the opening brace character ({}).
Commas to separate MarkupExtension parameters	Specifies whether commas are generated when you type more than one parameter in a markup extension.

Errors and Warnings

Use this setting to control when errors and warnings are automatically generated. For more information, see [XAML errors and warnings](#).

See also

- [XAML in WPF](#)

Options, Text Editor, XML, Formatting

10/18/2019 • 2 minutes to read • [Edit Online](#)

Use the **Formatting** options page to specify how elements and attributes are formatted in your XML documents. To access XML formatting options, choose **Tools > Options > Text Editor > XML**, and then choose **Formatting**.

Attributes

Preserve manual attribute formatting

Do not reformat attributes. This setting is the default.

NOTE

If the attributes are on multiple lines, the editor indents each line of attributes to match the indentation of the parent element.

Align attributes each on a separate line

Align the second and subsequent attributes vertically to match the indentation of the first attribute. The following XML text is an example of how the attributes would be aligned:

```
<item id = "123-A"
      name = "hammer"
      price = "9.95">
</item>
```

Auto Reformat

On paste from clipboard

Reformat XML text pasted from the clipboard.

On completion of end tag

Reformat the element when the end tag is completed.

Mixed Content

Format mixed content by default.

Attempt to reformat mixed content, except when the content is found in an `xml:space="preserve"` scope. This setting is the default.

If an element contains a mix of text and markup, the contents are considered to be mixed content. Following is an example of an element with mixed content.

```
<dir>c:\data\AlphaProject\
  <file readOnly="false">test1.txt</file>
  <file readOnly="false">test2.txt</file>
</dir>
```

See also

- [XML options - miscellaneous](#)
- [XML tools in Visual Studio](#)

Options, Text Editor, XML, Miscellaneous

10/18/2019 • 2 minutes to read • [Edit Online](#)

Use the **Miscellaneous** options page to change the autocompletion and schema settings for the XML Editor. To access miscellaneous XML options, choose **Tools > Options > Text Editor > XML**, and then choose **Miscellaneous**.

Auto Insert

Close tags

The text editor adds close tags when authoring XML elements. If an element start tag is selected, the editor inserts the matching close tag, including a matching namespace prefix. This check box is selected by default.

Attribute quotes

When authoring XML attributes, the editor inserts the `="` and `"` characters and positions the caret (^) inside the quotation marks. This check box is selected by default.

Namespace declarations

The editor automatically inserts namespace declarations wherever they are needed. This check box is selected by default.

Other markup (Comments, CDATA)

Comments, CDATA, DOCTYPE, processing instructions, and other markup is autocompleted. This check box is selected by default.

Network

Automatically download DTDs and schemas

Schemas and document type definitions (DTDs) are automatically downloaded from HTTP locations. This feature uses System.Net with autoproxy server detection enabled. This check box is selected by default.

Outlining

Enter outlining mode when files open

Turns on the outlining feature when a file is opened. This check box is selected by default.

Caching

Schemas

Specifies the location of the schema cache. The **Browse** button opens the current schema cache location in a new window. The default location is `%VsInstallDir%\xml\Schemas`.

See also

- [XML options - formatting](#)
- [XML tools in Visual Studio](#)

Options dialog box: Windows Forms Designer

10/18/2019 • 2 minutes to read • [Edit Online](#)

The Windows Forms Designer options page lets you set preferences for the grids and other features of the Windows Forms Designer in Visual Studio. Open the **Options** dialog box from the **Tools** menu.

Code Generation Settings

Optimized Code Generation

Enables optimized code generation. Some controls may not be compatible with this mode. For this change to take effect, Visual Studio must be closed and reopened.

High-DPI Support

DPI Scaling Notifications

Show a message in Windows Form Designer that can restart Visual Studio with 100% scaling. For more information, see [Disable DPI-awareness in Visual Studio](#).

Layout Settings

Default Grid Cell Size

Sets the spacing, in pixels, between horizontal and vertical gridlines on the designer. The default size is 8, 8. The maximum size is 200, 200.

Layout Mode

Specifies the alignment system to use for layout. You can choose either SnapToGrid or Snaplines.

Show Grid

Specifies whether designers display the sizing grid. By default, the grid is on.

Snap to Grid

Determines whether designers will snap objects and controls to the grid. In other words, the resizing and movement of elements on the designer are constrained to the GridSize increment when this feature is turned on. Having SnapToGrid turned on makes it easier to line up the various aspects of the user interface precisely but limits the freedom with which one can place controls. By default, SnapToGrid is turned on.

Object Bound Smart Tag Settings

Automatically Open Smart Tags

Determines whether controls and components display smart tags. Not all controls and components support smart tags.

Refactoring

Enable Refactoring on Rename

When set to `true`, a rename refactoring operation is performed when you rename a component from the Properties window or Document Outline window.

Toolbox

Automatically Populate Toolbox

Determines whether the Toolbox window is automatically populated with components and controls built by the project.

Options dialog box: Windows Forms Designer > Data UI Customization

10/18/2019 • 2 minutes to read • [Edit Online](#)

This dialog box defines which controls appear in the list of available controls for items in the Data Sources window. To open it, select **Tools** > **Options**, and then select **Windows Forms Designer** > **Data UI Customization**.

You can select a control from an item in the Data Sources window prior to dragging it onto the form in a Windows Forms app. Available controls are determined by the data type of the item. Each data type has a list of valid associated controls as defined in this dialog box, including a default control. When you drag an item from the Data Sources window onto a form without selecting a control, the default control for the data type of the selected item is added to the form.

Customize the list of associated controls by selecting and clearing the check boxes of available controls for each data type. To add a control to the list, add a control that implements either the [DefaultBindingPropertyAttribute](#) or [ComplexBindingPropertiesAttribute](#) data-binding attribute to the Toolbox. The control will then appear in the list of controls for the data type. For more information, see [How to: Add Custom Controls to the Data Sources window](#).

Data type

Displays a list of types with which you associate controls. Tables are represented as the [\[List\]](#) data type. Columns are represented as the actual data type of the column in the underlying data store.

Associated controls

Displays a list of controls that are associated with the selected data type. Select or clear the checkbox next to the control to associate or disassociate it. Selected controls appear in Data Sources window for a database column that's bound to the associated data type.

Set Default

Assigns the selected control type to be the default for the selected data type. The default control appears as the first selection in the shortcut menu for a database column in the Data Sources window. When you drag an item from the Data Sources window onto a form without selecting a control, the default control for the data type of the selected item is added to the form.

Only one control type can be assigned as the default for a data type.

Clear Default

Removes the designation of a control as the default for the selected data type. If there is no default for the selected data type, [\[None\]](#) appears as the first selection in the shortcut menu for a database column of that type.

XAML Designer options page

10/18/2019 • 4 minutes to read • [Edit Online](#)

Use the **XAML Designer** options page to specify how elements and attributes are formatted in your XAML documents. To open this page, choose the **Tools** menu and then choose **Options**. To access the **XAML Designer** property page, choose the **XAML Designer** node. Settings for the XAML Designer are applied when you open the document. So, if you make changes to the settings, you need to close and then reopen Visual Studio to see the changes.

NOTE

The dialog boxes and menu commands you see might differ from those described in Help depending on your active settings or edition. To change your settings, choose **Import and Export Settings** on the **Tools** menu. For more information, see [Reset settings](#).

Enable XAML Designer

When selected, this setting enables the XAML Designer. The XAML Designer provides a visual work area for you to edit XAML documents. Certain functionality in Visual Studio, such as IntelliSense for resources and databinding, require the XAML Designer to be enabled.

The following settings apply only when XAML Designer is enabled. If you change this option, you will need to restart Visual Studio for the setting to take effect.

Default document view

Use this setting to control whether Design view appears when XAML documents are loaded.

Source View	Specifies whether only XAML source appears in the XAML view. This is useful when loading large documents.
Design View	Specifies whether only a visual XAML Designer appears in the XAML view.
Split View	Specifies whether both the visual XAML Designer and the XAML source appear next to one another in the XAML view (location based on the Split Orientation setting).

Split Orientation

Use this setting to control when and how the XAML Designer appears when editing a XAML document. These settings apply only when **Default document view** is set to **Split View**.

Vertical	XAML source appears on the left side of the XAML view, and the XAML Designer appears on the other side.
-----------------	---

Horizontal	The XAML Designer appears on the top of the XAML view, and the XAML source appears below it.
Default	The XAML document uses the split orientation recommended for the platform targeted by the document's project. For most platforms this is equivalent to Horizontal .

Zoom by using

Use this setting to determine how zoom works when editing a XAML document.

Mouse wheel	Zoom in the XAML Designer by scrolling the mouse wheel.
Ctrl + mouse wheel	Zoom in the XAML Designer by pressing the Ctrl key while scrolling the mouse wheel.
Alt + mouse wheel	Zoom in the XAML Designer by pressing the Alt key while scrolling the mouse wheel.

These settings determine Designer behavior when editing a XAML document.

Automatically name interactive elements on creation	Specifies whether a default name is provided for a new interactive element when you add one to the Designer.
Automatically insert layout properties on element creation	Specifies whether layout properties are provided for a new element when you add one to the Designer. Layout properties are those that impact the layout of a control, for example, Margin and VerticalAlignment. The following XAML shows how a Button is created with and without this option selected: <div style="border: 1px solid black; padding: 5px; margin-top: 5px;"> <Button Content="Button" HorizontalAlignment="Left" Margin="245,56,0,0" Grid.Row="1" VerticalAlignment="Top" Width="75"/> <Button Content="Button" Grid.Row="1"/> </div>
Use quadrant based layout	Specifies whether the currently selected control aligns to the nearest edges of the parent container. If this checkbox is cleared, control alignments do not change during a move or create operation.
Automatically populate toolbox items	Specifies whether user controls and custom controls in the current solution are shown in the Toolbox automatically.

Settings (Blend only)

Use these options to determine settings when editing XAML files using Blend.

Zoom by using	Zoom in the XAML Designer by scrolling the mouse wheel, or by pressing the Ctrl or Alt key while scrolling the mouse wheel.
----------------------	---

Type units	Specifies whether measurements on the designer are based on points or pixels. Because Universal Windows Apps don't support points, units are automatically converted to pixels if Points is selected.
-------------------	--

Artboard (Blend only)

Use these settings to determine XAML Designer behavior when editing XAML documents in Blend.

Snapping

Show snap grid	When this option is selected, gridlines appear in the designer to help you align controls. Controls added to the designer snap to these gridlines when the Snap to gridlines option is selected.
Snap to gridlines	When controls are added or moved around the designer, they snap to the gridlines.
Gridline spacing	Specifies the spacing between the gridlines in either pixels or points (as determined by the Type units setting).
Snap to snaplines	Specifies whether controls snap to snaplines.
Default margin	When Snap to snaplines is enabled, specifies the spacing between the control and the snaplines in either pixels or points (as determined by the Type units setting).
Default padding	When Snap to snaplines is enabled, specifies the extra spacing between the control and the snaplines in either pixels or points (as determined by the Type units setting).

Animation

Use this setting to determine whether a warning appears when dependent (non-accelerated) animations are enabled in Blend.

Effects

Use these settings to determine whether effects are rendered when editing XAML files in the XAML Designer using Blend.

Render effects	Specifies whether effects render when editing XAML files in the XAML Designer using Blend.
Zoom threshold	Specifies the percentage of zoom in which effects render when the Render effects checkbox is selected. If you zoom beyond this setting, effects no longer render in the XAML Designer.

See also

- [XAML in WPF](#)
- [Walkthrough: My first WPF desktop application](#)

Output window

10/18/2019 • 2 minutes to read • [Edit Online](#)

The **Output** window displays status messages for various features in the integrated development environment (IDE). To open the **Output** window, on the menu bar, choose **View > Output**, or press **Ctrl+Alt+O**.

Toolbar

The following controls are shown in the toolbar of the **Output** window.

Show output from

Displays one or more output panes to view. Several panes of information might be available, depending on which tools in the IDE have used the **Output** window to deliver messages to the user.

Find Message in Code

Moves the insertion point in the code editor to the line that contains the selected build error.

Go to Previous Message

Changes the focus in the **Output** window to the previous build error and moves the insertion point in the code editor to the line that contains that build error.

Go to Next Message

Changes the focus in the **Output** window to the next build error and moves the insertion point in the code editor to the line that contains that build error.

Clear all

Clears all text from the **Output** pane.

Toggle Word Wrap

Turns the Word Wrap feature on and off in the **Output** pane. When Word Wrap is on, text in longer entries that extends beyond the viewing area is displayed on the following line.

Output pane

The **Output** pane selected in the **Show output from** list displays output from the source indicated.

Route messages to the Output window

To display the **Output** window whenever you build a project, in the **Options** dialog box, on the **Projects and Solutions > General** page, select **Show Output window when build starts**. Then, with a code file open for editing, choose **Go to Next Message** and **Go To Previous Message** on the **Output** window toolbar to select entries in the **Output** pane. As you do this, the insertion point in the code editor jumps to the line of code where the selected problem occurs.

Certain IDE features and commands invoked in the **Command window** deliver their output to the **Output** window. Output from external tools such as *.bat* and *.com* files, which is typically displayed in the command window, is routed to an **Output** pane when you select the **Use Output Window** option in **Manage external tools**. Many other kinds of messages can be displayed in **Output** panes as well. For example, when Transact-SQL syntax in a stored procedure is checked against a target database, the results are displayed in the **Output** window.

You can also program your own applications to write diagnostic messages at run time to an **Output** pane. To do

this, use members of the [Debug](#) class or [Trace](#) class in the [System.Diagnostics](#) namespace of the .NET API. Members of the [Debug](#) class display output when you build Debug configurations of your solution or project; members of the [Trace](#) class display output when you build either Debug or Release configurations. For more information, see [Diagnostic messages in the Output window](#).

In C++, you can create custom build steps and build events whose warnings and errors are displayed and counted in the **Output** pane. By pressing **F1** on a line of output, you can display an appropriate help topic. For more information, see [Format the output of a custom build step](#).

Scroll behavior

If you use autoscrolling in the **Output** window and then navigate by using the mouse or arrow keys, autoscrolling stops. To resume autoscrolling, press **Ctrl+End**.

See also

- [Diagnostic messages in the Output window](#)
- [How to: Control the Output window](#)
- [Compile and build](#)
- [Understand build configurations](#)
- [Class library overview](#)

Project Properties reference

10/18/2019 • 2 minutes to read • [Edit Online](#)

Learn more about how to configure and customize project properties.

Project Properties pages

TITLE	DESCRIPTION
Application Page, Project Designer (Visual Basic)	Use this page to specify application settings and properties for a Visual Basic project.
Application Page, Project Designer (C#)	Use this page to specify application settings and properties for a Visual C# project.
Build Events Page, Project Designer (C#)	Use this pane to specify build configuration instructions.
Build Page, Project Designer (C#)	Use this pane to specify build configuration properties for a Visual C# project.
Compile Page, Project Designer (Visual Basic)	Use this page to specify compilation properties for Visual Basic projects.
Debug Page, Project Designer	Use this page to specify debugging properties for a project.
Code Analysis, Project Designer	Use this page to configure the code analysis tool.
Publish Page, Project Designer	Use this page to configure properties for ClickOnce.
References Page, Project Designer (Visual Basic)	Use this page to manage references used by a project.
Security Page, Project Designer	Use this page to configure code access security settings for applications that are deployed by using ClickOnce deployment.
Signing Page, Project Designer	Use this page to sign application and deployment manifests, and sign the assembly.

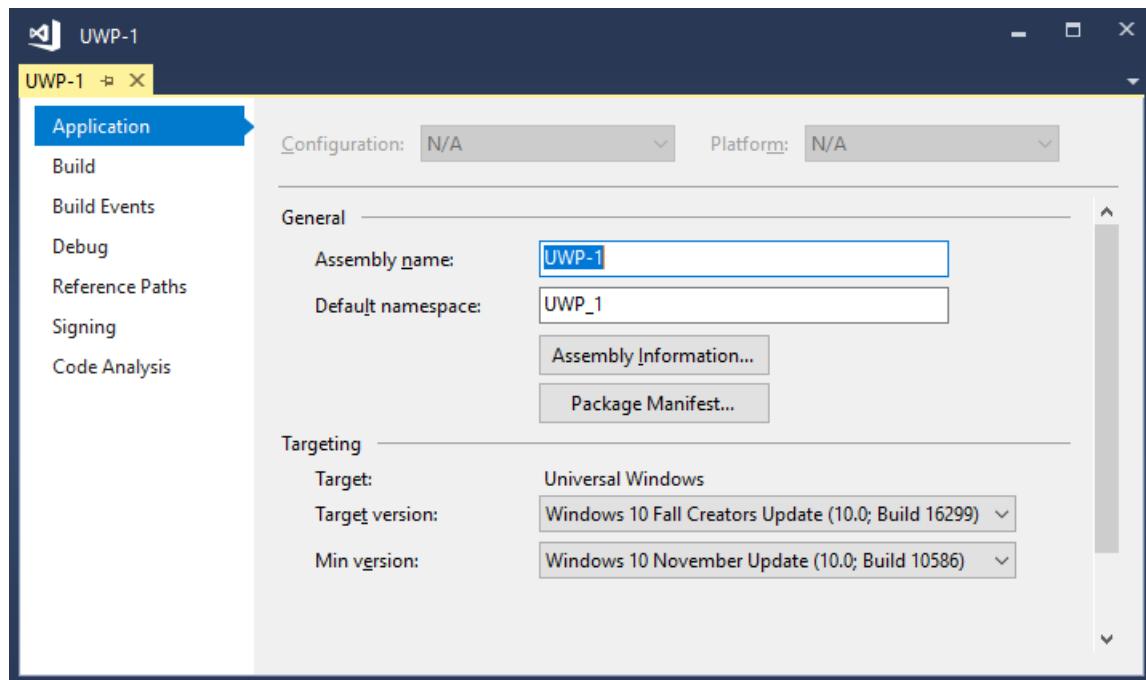
See also

- [Solutions and Projects](#)

Application property page (UWP projects)

10/18/2019 • 2 minutes to read • [Edit Online](#)

Use the **Application** property page to specify the Universal Windows Platform (UWP) project's assembly and package information, and target Windows 10 version.



To access the **Application** page, choose the project node in **Solution Explorer**. Then choose **Project > Properties** on the menu bar. The property pages open on the **Application** tab.

General section

Assembly name—Specifies the name of the output file that will hold the assembly manifest.

To access this property programmatically, see [AssemblyName](#).

Default namespace—Specifies the base namespace for files added to the project. For more information about namespaces, see [Namespaces \(C# programming guide\)](#), [Namespaces \(Visual Basic\)](#), or [Namespaces \(C++\)](#).

To access this property programmatically, see [RootNamespace](#).

Assembly Information—Choosing this button displays the [Assembly Information dialog box](#).

Package Manifest—Choosing this button opens the manifest designer. The manifest designer can also be accessed by choosing the *Package.appxmanifest* file in **Solution Explorer**. For more information, see [Configure a package with the manifest designer](#).

Targeting section

You can set the target version and minimum version of Windows 10 for your app by using the drop-down lists in this section. It is recommended that you target the latest version of Windows 10, and if you are developing an enterprise app, that you support an older minimum version too. For more information about which Windows 10 version to choose, see [Choose a UWP version](#).

For information about platform targeting in Visual Studio, see [Platform targeting](#).

See also

- [Create your first UWP app](#)
- [Choose a UWP version](#)

Application Page, Project Designer (Visual Basic)

10/18/2019 • 8 minutes to read • [Edit Online](#)

Use the **Application** page of the Project Designer to specify a project's application settings and properties.

To access the **Application** page, choose a project node (not the **Solution** node) in **Solution Explorer**. Then choose **Project > Properties** on the menu bar. When the **Project Designer** appears, select the **Application** tab.

NOTE

Your computer might show different names or locations for some of the Visual Studio user interface elements in this article.

You may be using a different edition of Visual Studio or different environment settings. For more information, see

[Personalize the IDE](#).

General application settings

The following options enable you to configure general settings for an application.

Assembly name

Specifies the name of the output file that will contain the assembly manifest. If you change this property, the **Output Name** property also changes.

You can also specify the name of the output file from a command prompt by using the [/out \(Visual Basic\)](#) compiler switch.

For information about how to access this property programmatically, see [AssemblyName](#).

Root namespace

Specifies the base namespace for all files in the project. For example, if you set the **Root Namespace** to

`Project1` and you have a `Class1` outside of any namespace in your code, its namespace would be

`Project1.Class1`. If you have a `Class2` in a namespace `Order` in code, its namespace would be

`Project1.Order.Class2`.

If you clear the **Root Namespace**, you can specify the namespace structure of your project in code.

NOTE

If you use the `Global` keyword in a [Namespace Statement](#), you can define a namespace out of the root namespace of your project. If you clear the **Root Namespace**, `Global` becomes the top-level namespace, which removes the need for the `Global` keyword in a `Namespace` statement. For more information, see "Global Keyword in Namespace Statements" in [Namespaces in Visual Basic](#).

For information about how to create namespaces in your code, see [Namespace Statement](#).

For more information about the root namespace property, see [/rootnamespace](#).

For information about how to access this property programmatically, see [RootNamespace](#).

Target framework (all configurations)

Specifies the version of .NET that the application targets. This option can have different values depending on which versions of .NET are installed on your computer.

For .NET Framework projects, the default value matches the target framework that you specified when you created the project.

NOTE

The prerequisite packages that are listed in the [Prerequisites Dialog Box](#) are set automatically when you open the dialog box for the first time. If you subsequently change the project's target framework, you must specify the prerequisites manually to match the new target framework.

For more information, see [Framework targeting overview](#).

Application type

Specifies the type of application to build. The values are different depending on the project type. For example, for a **Windows Forms App** project, you can specify **Windows Forms Application**, **Class Library**, **Console Application**, **Windows Service**, or **Web Control Library**.

For a web application project, you must specify **Class Library**.

For more information about the **Application type** property, see [/target \(Visual Basic\)](#). For information about how to access that property programmatically, see [OutputType](#).

Auto-generate binding redirects

Binding redirects are added to your project if your app or its components reference more than one version of the same assembly. If you want to manually define binding redirects in the project file, deselect **Auto-generate binding redirects**.

For more information about redirection, see [Redirecting assembly versions](#).

Startup form / Startup object / Startup URI

Specifies the application's startup form or entry point.

If **Enable application framework** is selected (the default), this list is titled **Startup form** and shows only forms because the application framework supports only startup forms, not objects.

If the project is a WPF Browser Application, this list is titled **Startup URI**, and the default is **Page1.xaml**. The **Startup URI** list enables you to specify the user interface resource (a XAML element) that the application displays when the application starts. For more information, see [StartupUri](#).

If **Enable application framework** is cleared, this list becomes **Startup object** and shows both forms and classes or modules with a `Sub Main`.

Startup object defines the entry point to be called when the application loads. Generally this is set to either the main form in your application or to the `Sub Main` procedure that should run when the application starts. Because class libraries do not have an entry point, their only option for this property is **(None)**. For more information, see [/main](#). To access this property programmatically, see [StartupObject](#).

Icon

Sets the .ico file that you want to use as your program icon. Select **<Browse...>** to browse for an existing graphic. See [/win32icon](#) (or [/win32icon \(C# Compiler Options\)](#)) for more information. To access this property programmatically, see [ApplicationIcon](#).

Assembly Information

Click this button to display the [Assembly Information Dialog Box](#).

Enable application framework

Specifies whether a project will use the application framework. The setting of this option affects the options

available in **Startup form/Startup object**.

If this check box is selected, your application uses the standard `Sub Main`. Selecting this check box enables the features in the **Windows application framework properties** section, and also requires you to select a startup form.

If this check box is cleared, your application uses the custom `Sub Main` that you specified in **Startup form**. In this case you can specify either a startup object (a custom `Sub Main` in a method or a class) or a form. Also, the options in the **Windows application framework properties** section become unavailable.

View Windows Settings

Click this button to generate and open the `app.manifest` file. Visual Studio uses this file to generate manifest data for the application. Then set the UAC requested execution level by modifying the `<requestedExecutionLevel>` tag in `app.manifest` as follows:

```
<requestedExecutionLevel level="asInvoker" />
```

ClickOnce works with a level of `asInvoker` or in virtualized mode (no manifest generation). To specify virtualized mode, remove the entire tag from `app.manifest`.

For more information about manifest generation, see [ClickOnce Deployment on Windows Vista](#).

Windows application framework properties

The following settings are available in the **Windows application framework properties** section. These options are available only if the **Enable application framework** check box is selected.

TIP

The section following this one describes **Windows application framework properties** settings specific to Windows Presentation Foundation (WPF) apps.

Enable XP visual styles

Enables or disables the Windows XP visual styles, also known as *Windows XP Themes*. Windows XP visual styles enable, for example, controls with rounded corners and dynamic colors. The default is enabled.

Make single instance application

Select this check box to prevent users from running multiple instances of the application. The default setting for this check box is *cleared*, which allows multiple instances of the application to be run. For more information, see the [StartupNextInstance](#) event.

Save My.Settings on Shutdown

Select this check box to specify that the application's `My.Settings` settings are saved when users shut down their computers. The default setting is enabled. If this option is disabled, you can save application settings manually by calling `My.Settings.Save`.

Authentication mode

Select **Windows** (the default) to specify the use of Windows authentication to identify the currently logged-on user. You can retrieve this information at run time by using the `My.User` object. Select **Application-defined** if you will provide your own code to authenticate users instead of using the default Windows authentication methods.

Shutdown mode

Select **When startup form closes** (the default) to specify that the application exit when the form set as the startup form closes, even if other forms are open. Select **When last form closes** to specify that the application

exit when the last form is closed or when `My.Application.Exit` or the `End` statement is called explicitly.

Select **On explicit shutdown** to specify that the application exit when you explicitly call `Shutdown`.

Select **On last window close** to specify that the application exit when the last window closes or when you explicitly call `Shutdown`. This is the default setting.

Select **On main window close** to specify that the application exit when the main window closes or when you explicitly call `Shutdown`.

Splash screen

Select the form that you want to use as a splash screen. You must have previously created a splash screen by using a form or a template. The default is **(None)**.

View Application Events

Click this button to display an events code file in which you can write events for the application framework events `Startup`, `Shutdown`, `UnhandledException`, `StartupNextInstance` and `NetworkAvailabilityChanged`. You can also override certain application framework methods. For example, you can change the display behavior of the splash screen by overriding `OnInitialize`.

Windows application framework properties for Windows Presentation Foundation (WPF) apps

The following settings are available in the **Windows application framework properties** section when the project is a Windows Presentation Foundation (WPF) app. These options are available only if the **Enable application framework** check box is selected. The options listed in this table are available only for WPF or WPF browser applications. They are not available for WPF User Control or Custom Control libraries.

Shutdown mode

This property is applicable only to Windows Presentation Foundation (WPF) applications.

Select **On explicit shutdown** to specify that the application exit when you explicitly call `Shutdown`.

Select **On last window close** to specify that the application exit when the last window closes or when you explicitly call `Shutdown`. This is the default setting.

Select **On main window close** to specify that the application exit when the main window closes or when you explicitly call `Shutdown`.

For more information about using this setting, see [Shutdown](#)

Edit XAML

This button opens the application definition file (`Application.xaml`) in the XAML editor. When you click this button, `Application.xaml` opens at the application definition node. You might have to edit this file to perform certain tasks, such as defining resources. If the application definition file does not exist, the Project Designer creates one.

View Application Events

This button opens the `Application` class file (`Application.xaml.vb`) in a code editor. If the file does not exist, the Project Designer creates one with the appropriate class name and namespace.

The `Application` object raises events when certain application state changes occur (for example, on application startup or shutdown). For a full list of the events that this class exposes, see [Application](#). These events are handled in the user code section of the `Application` partial class.

Assembly Information dialog box

10/18/2019 • 2 minutes to read • [Edit Online](#)

The Assembly Information dialog box is used to specify the values of the .NET Framework global assembly attributes, which are stored in the AssemblyInfo file created automatically with your project. In Solution Explorer, the AssemblyInfo file is located in the **My Project** node for Visual Basic projects (click **Show All files** to view it). For C# projects, it's located under **Properties**. For more information, see [Attributes \(C#\)](#).

To access this dialog box, select a project node in **Solution Explorer**, and then, on the **Project** menu, select **Properties**. On the **Application** page, select the **Assembly Information** button.

UIElement list

Title

Specifies a title for the assembly manifest. Corresponds to [AssemblyTitleAttribute](#).

Description

Specifies an optional description for the assembly manifest. Corresponds to [AssemblyDescriptionAttribute](#).

Company

Specifies a company name for the assembly manifest. Corresponds to [AssemblyCompanyAttribute](#).

You can set or change the default value for Company in the registry. Look for the **RegisteredOrganization** value under the **Computer\HKEY_LOCAL_MACHINE\SOFTWARE\WOW6432Node\Microsoft\Windows NT\CurrentVersion** or **Computer\HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion** key, depending on your version of Windows.

Product

Specifies a product name for the assembly manifest. Corresponds to [AssemblyProductAttribute](#).

Copyright

Specifies a copyright notice for the assembly manifest. Corresponds to [AssemblyCopyrightAttribute](#).

Trademark

Specifies a trademark for the assembly manifest. Corresponds to [AssemblyTrademarkAttribute](#).

Assembly Version

Specifies the version of the assembly. Corresponds to [AssemblyVersionAttribute](#).

File Version

Specifies a version number that instructs the compiler to use a specific version for the Win32 file version resource. Corresponds to [AssemblyFileVersionAttribute](#).

GUID

A unique GUID that identifies the assembly. When you create a project, Visual Studio generates a GUID for the assembly. Corresponds to [Guid](#).

Neutral Language

Specifies which culture the assembly supports. Corresponds to [NeutralResourcesLanguageAttribute](#). The default is **(None)**.

Make assembly COM-Visible

Specifies whether types in the assembly will be available to COM. Corresponds to [ComVisibleAttribute](#).

NOTE

For more information on setting these properties when generating a NuGet package in a .NET Framework class library, see [Configure project properties for the package](#).

See also

- [Application Page, Project Designer \(Visual Basic\)](#)
- [Attributes](#)

Application Page, Project Designer (C#)

10/18/2019 • 4 minutes to read • [Edit Online](#)

Use the **Application** page of the **Project Designer** to specify the project's application settings and properties.

To access the **Application** page, choose a project node (not the **Solution** node) in **Solution Explorer**. Then choose **Project > Properties** on the menu bar. When the **Project Designer** appears, click the **Application** tab.

NOTE

Your computer might show different names or locations for some of the Visual Studio user interface elements in this article.

You may be using a different edition of Visual Studio or different environment settings. For more information, see [Personalize the IDE](#).

General Application Settings

The following options enable you to configure general settings for the application.

Assembly name

Specifies the name of the output file that will hold the assembly manifest. Changing this property also changes the **Output Name** property.

You can also make this change from the command line by using [/out \(C# Compiler Options\)](#).

To access this property programmatically, see [AssemblyName](#).

Default namespace

Specifies the base namespace for files added to the project.

See [namespace](#) for more information about creating namespaces in your code.

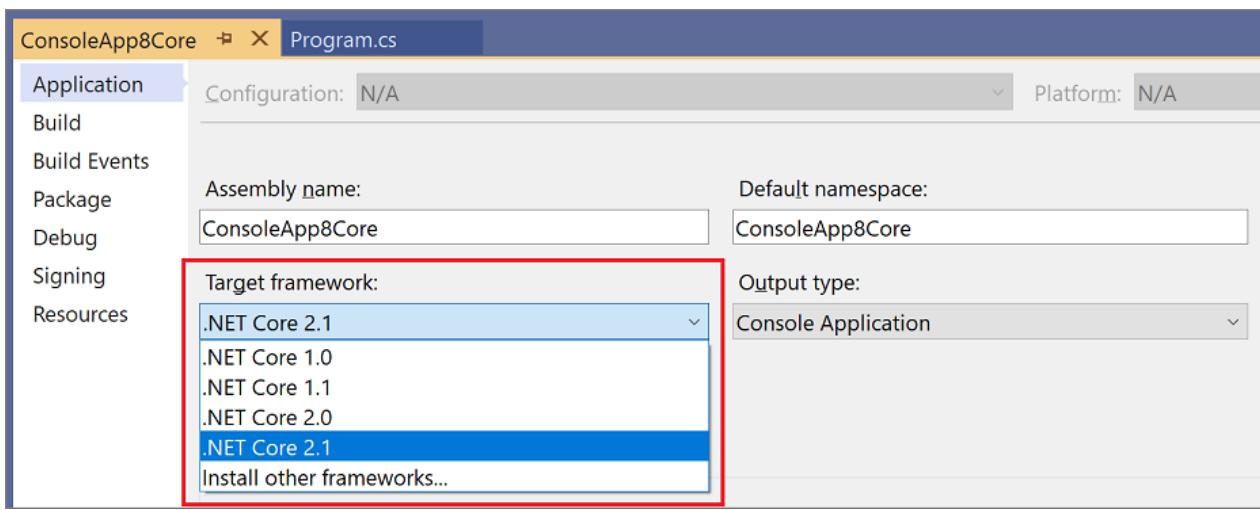
To access this property programmatically, see [RootNamespace](#).

Target Framework

Specifies the version of .NET that the application targets. This option can have different values depending on which versions of .NET are installed on your computer.

For .NET Framework projects, the default value matches the target framework that you specified when you created the project.

For a project that targets .NET Core, the available versions may appear as follows:



NOTE

The prerequisite packages listed in the [Prerequisites Dialog Box](#) are set automatically the first time that you open the dialog box. If you subsequently change the project's target framework, you must select the prerequisites manually to match the new target framework.

For more information, see [Framework targeting overview](#).

Output type

Specifies the type of application to build. The values are different depending on the project type. For example, for a **Console App** project, you can specify **Windows Application**, **Console Application**, or **Class Library** as the output type.

For a web application project, you must specify **Class Library**.

For more information about the **Output type** property, see [/target \(C# Compiler Options\)](#).

For information about how to access this property programmatically, see [OutputType](#).

Auto-generate binding redirects

Binding redirects are added to your project if your app or its components reference more than one version of the same assembly. If you want to manually define binding redirects in the project file, deselect **Auto-generate binding redirects**.

For more information about redirection, see [Redirecting assembly versions](#).

Startup object

Defines the entry point to be called when the application loads. Generally this is set either to the main form in your application or to the `Main` procedure that should run when the application starts. Because class libraries do not have an entry point, their only option for this property is **(Not set)**.

By default, in a WPF app project, this option is set to **(Not set)**. The other option is `[projectname].App`. In a WPF project, you must set the startup URI to load a UI resource when the application starts. To do this, open the `Application.xaml` file in your project and set the `StartupUri` property to a `.xaml` file in your project, such as `Window1.xaml`. For a list of acceptable root elements, see [StartupUri](#). You must also define a `public static void Main()` method in a class in the project. This class will appear in the **Startup object** list as `ProjectName.ClassName`. You can then select the class as the startup object.

See [/main \(C# Compiler Options\)](#) for more information. To access this property programmatically, see [StartupObject](#).

Assembly Information

This button opens the [Assembly Information](#) dialog box.

Resources

The **Resources** options help you configure resource settings for your app.

Icon and manifest

By default, this radio button is selected and the **Icon** and **Manifest** options are enabled. This enables you to select your own icon or to select different manifest generation options. Leave this radio button selected unless you're providing a resource file for the project.

Icon

Sets the .ico file that you want to use as your program icon. Click **Browse** to browse for an existing graphic, or type the name of the file that you want. See [/win32icon \(C# Compiler Options\)](#) for more information.

To access this property programmatically, see [ApplicationIcon](#).

For information about creating an icon, see [Image editor for icons](#).

Manifest

Selects a manifest generation option when the application runs on Windows Vista under User Account Control (UAC). This option can have the following values:

- **Embed manifest with default settings.** Supports the typical manner in which Visual Studio operates on Windows Vista, which is to embed security information in the application's executable file, specifying that `requestedExecutionLevel` be `AsInvoker`. This is the default option.
- **Create application without a manifest.** This method is known as *virtualization*. Use this option for compatibility with earlier applications.
- **Properties\app.manifest.** This option is required for applications deployed by ClickOnce or Registration-Free COM. If you publish an application by using ClickOnce deployment, **Manifest** is automatically set to this option.

Resource file

Select this radio button when you're providing a resource file for the project. Selecting this option disables the **Icon** and **Manifest** options.

Enter a path name or use the Browse button (...) to add a Win32 resource file to the project.

For more information, see [Create resource files for .NET apps](#).

Build Events Page, Project Designer (C#)

11/7/2019 • 2 minutes to read • [Edit Online](#)

Use the **Build Events** page of the **Project Designer** to specify build configuration instructions. You can also specify the conditions under which any post-build events are run. For more information, see [How to: Specify Build Events \(C#\)](#) and [How to: Specify Build Events \(Visual Basic\)](#).

UIElement List

Configuration

This control is not editable in this page. For a description of this control, see [Build Page, Project Designer \(C#\)](#).

Platform

This control is not editable on this page. For a description of this control, see [Build Page, Project Designer \(C#\)](#).

Pre-build event command line

Specifies any commands to execute before the build starts. To type long commands, click **Edit Pre-build** to display the [Pre-build Event/Post-build Event Command Line Dialog Box](#).

NOTE

Pre-build events do not run if the project is up to date and no build is triggered.

Post-build event command line

Specifies any commands to execute after the build ends. To type long commands, click **Edit Post-build** to display the [Pre-build Event/Post-build Event Command Line Dialog Box](#).

NOTE

Add a `call` statement before all post-build commands that run .bat files. For example, `call C:\MyFile.bat` or `call C:\MyFile.bat call C:\MyFile2.bat`.

Run the post-build event

Specifies the following conditions for the post-build event to run, as shown in the following table.

OPTION	RESULT
Always	Post-build event will run regardless of whether the build succeeds.
On successful build	Post-build event will run if the build succeeds. Thus, the event will run even for a project that is up-to-date, as long as the build succeeds.
When the build updates the project output	Post-build event will only run when the compiler's output file (.exe or .dll) is different than the previous compiler output file. Thus, a post-build event is not run if a project is up-to-date.

In the project file

In earlier versions of Visual Studio, when you change the **PreBuildEvent** or **PostBuildEvent** setting in the IDE, Visual Studio adds a **PreBuildEvent** or **PostBuildEvent** property to the project file. So for example, if your **PreBuildEvent** command line setting in the IDE is follows:

```
"$(ProjectDir)PreBuildEvent.bat" "$(ProjectDir)..\" "$(ProjectDir)" "$(TargetDir)"
```

then the project file setting is:

```
<PropertyGroup>
    <PreBuildEvent>"$(ProjectDir)PreBuildEvent.bat" "$(ProjectDir)..\" "$(ProjectDir)" "$(TargetDir)" />
</PropertyGroup>
```

For .NET Core projects, Visual Studio 2019 (and Visual Studio 2017 in more recent updates) adds an MSBuild target named **PreBuild** or **PostBuild** for **PreBuildEvent** and **PostBuildEvent** settings. These targets use the **BeforeTargets** and **AfterTargets** attributes, which MSBuild recognizes. For example, for the preceding example, Visual Studio now generates the following code:

```
<Target Name="PreBuild" BeforeTargets="PreBuildEvent">
    <Exec Command="$(ProjectDir)PreBuildEvent.bat" ;$(ProjectDir)..\" ;
    $(ProjectDir); ;$(TargetDir); />
</Target>
```

For a post-build event, use the name **PostBuild** and set the attribute **AfterTargets** to **PostBuildEvent**.

```
<Target Name="PostBuild" AfterTargets="PostBuildEvent">
    <Exec Command="echo Output written to $(TargetDir)" />
</Target>
```

NOTE

These project file changes were made to support SDK-style projects. If you are migrating a project file from the old format to the SDK-style format manually, you should delete the **PreBuildEvent** and **PostBuildEvent** properties and replace them with **PreBuild** and **PostBuild** targets as shown in the preceding code. To find out how to tell if your project is an SDK-style project, see [Check project format](#).

See also

- [How to: Specify Build Events \(Visual Basic\)](#)
- [How to: Specify Build Events \(C#\)](#)
- [Project Properties Reference](#)
- [Compiling and Building](#)

Pre-build event/post-build event command line dialog box

7/24/2019 • 2 minutes to read • [Edit Online](#)

You can type pre- or post-build events for the [Build Events Page](#), [Project Designer \(C#\)](#) directly in the edit box, or you can select pre- and post-build macros from a list of available macros.

NOTE

Pre-build events do not run if the project is up to date and no build is triggered.

UI Element List

Command line edit box

Contains the events to run either for pre-build or post-build.

NOTE

Add a `call` statement before all post-build commands that run .bat files. For example, `call C:\MyFile.bat` or
`call C:\MyFile.bat call C:\MyFile2.bat`.

Macros

Expands the edit box to display a list of macros to insert in the command-line edit box.

Macro table

Lists the available macros and its value. See Macros below for a description of each. You can select only one macro at a time to insert into the command-line edit box.

Insert

Inserts into the command line edit box the macro selected in the macro table.

Macros

You can use any of these macros to specify locations for files, or to get the actual name of the input file in the case of multiple selections. These macros are not case-sensitive.

MACRO	DESCRIPTION
<code>\$(ConfigurationName)</code>	The name of the current project configuration, for example, "Debug".
<code>\$(OutDir)</code>	Path to the output file directory, relative to the project directory. This resolves to the value for the Output Directory property. It includes the trailing backslash '\'.
<code>\$(DevEnvDir)</code>	The installation directory of Visual Studio (defined with drive and path); includes the trailing backslash '\'.

MACRO	DESCRIPTION
<code>\$(PlatformName)</code>	The name of the currently targeted platform. For example, "AnyCPU".
<code>\$(ProjectDir)</code>	The directory of the project (defined with drive and path); includes the trailing backslash '\'.
<code>\$(ProjectPath)</code>	The absolute path name of the project (defined with drive, path, base name, and file extension).
<code>\$(ProjectName)</code>	The base name of the project.
<code>\$(ProjectFileName)</code>	The file name of the project (defined with base name and file extension).
<code>\$(ProjectExt)</code>	The file extension of the project. It includes the '.' before the file extension.
<code>\$(SolutionDir)</code>	The directory of the solution (defined with drive and path); includes the trailing backslash '\'.
<code>\$(SolutionPath)</code>	The absolute path name of the solution (defined with drive, path, base name, and file extension).
<code>\$(SolutionName)</code>	The base name of the solution.
<code>\$(SolutionFileName)</code>	The file name of the solution (defined with base name and file extension).
<code>\$(SolutionExt)</code>	The file extension of the solution. It includes the '.' before the file extension.
<code>\$(TargetDir)</code>	The directory of the primary output file for the build (defined with drive and path). It includes the trailing backslash '\'.
<code>\$(TargetPath)</code>	The absolute path name of the primary output file for the build (defined with drive, path, base name, and file extension).
<code>\$(TargetName)</code>	The base name of the primary output file for the build.
<code>\$(TargetFileName)</code>	The file name of the primary output file for the build (defined as base name and file extension).
<code>\$(TargetExt)</code>	The file extension of the primary output file for the build. It includes the '.' before the file extension.

See also

- [Specifying Custom Build Events in Visual Studio](#)
- [Build Events Page, Project Designer \(C#\)](#)
- [How to: Specify Build Events \(Visual Basic\)](#)
- [How to: Specify Build Events \(C#\)](#)

Build Page, Project Designer (C#)

10/18/2019 • 5 minutes to read • [Edit Online](#)

Use the **Build** page of the **Project Designer** to specify the project's build configuration properties. This page applies to Visual C# projects only.

To access the **Build** page, choose a project node (not the **Solution** node) in **Solution Explorer**. Then choose **View, Property Pages** on the menu. When the Project Designer appears, choose the **Build** tab.

NOTE

Your computer might show different names or locations for some of the Visual Studio user interface elements in this article. You may be using a different edition of Visual Studio or different environment settings. For more information, see [Personalize the IDE](#).

Configuration and Platform

The following options enable you to select the configuration and platform to display or modify.

NOTE

With simplified build configurations, the project system determines whether to build a debug or release version. Therefore, these options are not displayed. For more information, see [How to: Set debug and release configurations](#).

Configuration

Specifies which configuration settings to display or modify. The settings can be **Active (Debug)** (this is the default), **Debug**, **Release**, or **All Configurations**.

Platform

Specifies which platform settings to display or modify. The default setting is **Active (Any CPU)**. You can change the active platform using the **Configuration Manager**. For more information, see [How to: Create and Edit Configurations](#).

General

The following options enable you to configure several C# compiler settings.

Conditional compilation symbols

Specifies symbols on which to perform conditional compilation. Separate symbols with a semi-colon (";"). For more information, see [/define \(C# Compiler Options\)](#).

Define DEBUG constant

Defines DEBUG as a symbol in all source code files in your app. Selecting this is equivalent to using the `/define:DEBUG` command-line option.

Define TRACE constant

Defines TRACE as a symbol in all source code files in your app. Selecting this is equivalent to using the `/define:TRACE` command-line option.

Platform Target

Specifies the processor to be targeted by the output file. Choose **x86** for any 32-bit Intel-compatible processor, choose **x64** for any 64-bit Intel-compatible processor, choose **ARM** for ARM processors, or choose **Any CPU** to specify that any processor is acceptable. **Any CPU** is the default value for projects, because it allows the application to run on the broadest range of hardware.

For more information, see [/platform \(C# Compiler Options\)](#).

Prefer 32-bit

If the **Prefer32-bit** check box is selected, the application runs as a 32-bit application on both 32-bit and 64-bit versions of Windows. If the check box is cleared, the application runs as a 32-bit application on 32-bit versions of Windows and as a 64-bit application on 64-bit versions of Windows.

If you run an application as a 64-bit application, the pointer size doubles, and compatibility problems can occur with other libraries that are exclusively 32-bit. It is useful to run a 64-bit application only if it needs more than 4 GB of memory or 64-bit instructions provide a significant performance improvement.

This check box is available only if all of the following conditions are true:

- On the **Build Page**, the **Platform target** list is set to **Any CPU**.
- On the **Application Page**, the **Output type** list specifies that the project is an application.
- On the **Application Page**, the **Target framework** list specifies the .NET Framework 4.5.

Allow unsafe code

Allows code that uses the `unsafe` keyword to compile. For more information, see [/unsafe \(C# Compiler Options\)](#).

Optimize code

Enable or disable optimizations performed by the compiler to make your output file smaller, faster, and more efficient. For more information, see [/optimize \(C# Compiler Options\)](#).

Errors and Warnings

The following settings are used to configure the error and warning options for the build process.

Warning level

Specifies the level to display for compiler warnings. For more information, see [/warn \(C# Compiler Options\)](#).

Suppress warnings

Blocks the compiler's ability to generate one or more warnings. Separate multiple warning numbers with a comma or semicolon. For more information, see [/nowarn \(C# Compiler Options\)](#).

Treat Warnings as Errors

The following settings are used to specify which warnings are treated as errors. Select one of the following options to indicate under what conditions to return an error when the build encounters a warning. For more information, see [/warnaserror \(C# Compiler Options\)](#).

None - Treats no warnings as errors.

All - Treats all warnings as errors.

Specific warnings - Treats the specified warnings as errors. Separate multiple warning numbers with a comma or semicolon.

TIP

If you don't want code analysis warnings to be treated as errors, see [Code analysis FAQ](#).

Output

The following settings are used to configure the output options for the build process.

Output path

Specifies the location of the output files for this project's configuration. Enter the path of the build output in this box, or choose the **Browse** button to specify a path. The path is relative; if you enter an absolute path, it will be saved as relative. The default path is bin\Debug or bin\Release\.

With simplified build configurations, the project system determines whether to build a debug or release version. The **Build** command from the **Debug** menu (F5) will put the build in the debug location regardless of the **Output path** you specify. However, the **Build** command from the **Build** menu puts it in the location you specify. For more information, see [Understanding Build Configurations](#).

XML documentation file

Specifies the name of a file into which documentation comments will be processed. For more information, see [/doc \(C# Compiler Options\)](#).

Register for COM interop

Indicates that your managed application will expose a COM object (a COM callable wrapper) that allows a COM object to interact with your managed application. The **Output type** property in the [Application page](#) of the **Project Designer** for this application must be set to **Class Library** in order for the **Register for COM interop** property to be available. For an example class that you might include in your Visual C# application and expose as a COM object, see [Example COM Class](#).

Generate serialization assembly

Specifies whether the compiler will use the XML Serializer Generator Tool (Sgen.exe) to create XML serialization assemblies. Serialization assemblies can improve the startup performance of [XmlSerializer](#) if you have used that class to serialize types in your code. By default, this option is set to **Auto**, which specifies that serialization assemblies be generated only if you have used [XmlSerializer](#) to encode types in your code to XML. **Off** specifies that serialization assemblies never be generated, regardless of whether your code uses [XmlSerializer](#). **On** specifies that serialization assemblies always be generated. Serialization assemblies are named `TypeName.XMLSerializers.dll`. For more information, see [XML Serializer Generator Tool \(Sgen.exe\)](#).

Advanced

Click to display the [Advanced Build Settings Dialog Box \(C#\)](#) dialog box.

See also

- [Project Properties Reference](#)
- [C# Compiler Options](#)

Advanced Build Settings dialog box (C#)

10/21/2019 • 2 minutes to read • [Edit Online](#)

Use the **Advanced Build Settings** dialog box of the **Project Designer** to specify the project's advanced build configuration properties. This dialog box applies to C# projects only.

General

The following options enable you to set general advanced settings.

Language Version

Links to [/langversion \(C# compiler options\)](#), which provides information about how a default language version is chosen based on a project's target framework.

Specifies the version of the language to use. The feature set is different in each version, so this option can be used to force the compiler to allow only a subset of the implemented features, or to enable only those features compatible with an existing standard.

The default value is C# 7.0.

Internal Compiler Error Reporting

Specifies whether to report compiler errors to Microsoft. If set to **prompt** (the default), you will receive a prompt if an internal compiler error occurs, giving you the option of sending an error report electronically to Microsoft. If set to **send**, an error report will be sent automatically. If set to **queue**, error reports will be queued. If set to **none**, the error will be reported only in the compiler's text output. For more information, see [/errorreport \(C# Compiler Options\)](#).

Check for arithmetic overflow/underflow

Specifies whether an integer arithmetic statement that is not in the scope of the **checked** or **unchecked** keywords and that results in a value outside the range of the data type will cause a run-time exception. For more information, see [/checked \(C# Compiler Options\)](#).

Do not reference mscorelib.dll

Specifies whether mscorelib.dll will be imported into your program, defining the entire **System** namespace. Check this box if you want to define or create your own **System** namespace and objects. For more information, see [/nostdlib \(C# Compiler Options\)](#).

Output

The following options enable you to specify advanced output options.

Debug Information

Specifies the type of debugging information generated by the compiler. For information on how to configure the debug performance of an application, see [Making an Image Easier to Debug](#). This setting has the following options:

- **none**

Specifies that no debugging information will be generated.

- **full**

Enables attaching a debugger to the running program.

- **pdbonly**

Allows source code debugging when the program is started in the debugger but will only display assembler when the running program is attached to the debugger.

- **portable**

Produces a .PDB file, a non-platform-specific, portable symbol file that provides other tools, especially debuggers, information about what is in the main executable file and how it was produced. See [Portable PDB](#) for more information.

- **embedded**

Embeds portable symbol information into the assembly. No external .PDB file is produced.

For more information, see [/debug \(C# Compiler Options\)](#).

File Alignment

Specifies the size of sections in the output file. Valid values are **512**, **1024**, **2048**, **4096**, and **8192**. These values are measured in bytes. Each section will be aligned on a boundary that is a multiple of this value, affecting the size of the output file. For more information, see [/filealign \(C# Compiler Options\)](#).

Library Base Address

Specifies the preferred base address at which to load a DLL. The default base address for a DLL is set by the .NET Framework common language runtime. For more information, see [/baseaddress \(C# Compiler Options\)](#).

See also

- [C# compiler options](#)
- [Build page, Project Designer \(C#\)](#)

Code Analysis, Project Designer

10/18/2019 • 2 minutes to read • [Edit Online](#)

Contains the code analysis tool that you can opt to run on your code. The tool reports information about your assemblies, such as violations of the programming and design rules set forth in the Microsoft .NET Design Guidelines.

UIElement list

Enable Code Analysis

Enables or disables code analysis for your project.

Design Rules

Enables or disables the design rules. You can also expand this entry to enable or disable individual rules.

Globalization Rules

Enables or disables the globalization rules. You can also expand this entry to enable or disable individual rules.

Interoperability Rules

Enables or disables the interoperability rules. You can also expand this entry to enable or disable individual rules.

Maintainability Rules

Enables or disables the maintainability rules. You can also expand this entry to enable or disable individual rules.

Mobility Rules

Enables or disables the mobility rules. You can also expand this entry to enable or disable individual rules.

Naming Rules

Enables or disables the naming rules. You can also expand this entry to enable or disable individual rules.

Performance Rules

Enables or disables the performance rules. You can also expand this entry to enable or disable individual rules.

Portability Rules

Enables or disables the portability rules. You can also expand this entry to enable or disable individual rules.

Reliability Rules

Enables or disables the reliability rules. You can also expand this entry to enable or disable individual rules.

Security Rules

Enables or disables the security rules. You can also expand this entry to enable or disable individual rules.

Usage Rules

Enables or disables the usage rules. You can also expand this entry to enable or disable individual rules.

See also

- [Code Analysis for Managed Code Warnings](#)
- [Code Analysis for Managed Code Overview](#)
- [Walkthrough: Analyzing Managed Code for Code Defects](#)

Compile Page, Project Designer (Visual Basic)

10/18/2019 • 8 minutes to read • [Edit Online](#)

Use the **Compile** page of the Project Designer to specify compilation instructions. You can also specify advanced compiler options and pre-build or post-build events on this page.

To access the **Compile** page, choose a project node (not the **Solution** node) in **Solution Explorer**. Then choose **Project, Properties** on the menu bar. When the Project Designer appears, click the **Compile** tab.

NOTE

Your computer might show different names or locations for some of the Visual Studio user interface elements in this article. You may be using a different edition of Visual Studio or different environment settings. For more information, see [Personalize the IDE](#).

Configuration and Platform

The following settings enable you to select the configuration and platform to display or modify.

NOTE

With simplified build configurations, the project system determines whether to build a debug or release version. Therefore, the **Configuration** and **Platform** lists are not displayed.

Configuration

Specifies which configuration settings to display or modify. The settings are **Debug** (default), **Release**, or **All Configurations**. For more information, see [Understanding Build Configurations](#) and [How to: Create and Edit Configurations](#).

Platform

Specifies which platform settings to display or modify. You can specify **Any CPU** (default), **x64**, or **x86**.

Compiler Configuration Options

The following settings enable you to set the compiler configuration options.

Build output path

Specifies the location of the output files for this project's configuration. Type the path of the build output in this box, or click the **Browse** button to select a path. Note that the path is relative; if you enter an absolute path, it will be saved as relative. The default path is bin\Debug\ or bin\Release\.

With simplified build configurations, the project system determines whether to build a debug or release version. The **Build** command from the **Debug** menu (F5) will put the build in the debug location regardless of the **Output path** you specify. However, the **Build** command from the **Build** menu puts it in the location you specify.

Option explicit

Specifies whether to allow implicit declaration of variables. Select **On** to require explicit declaration of variables. This causes the compiler to report errors if variables are not declared before they are used. Select **Off** to allow

implicit declaration of variables.

This setting corresponds to the [/optionexplicit](#) compiler option.

If a source code file contains an [Option Explicit Statement](#), the [On](#) or [off](#) value in the statement overrides the **Option Explicit** setting on the **Compile page**.

When you create a new project, the **Option Explicit** setting on the **Compile page** is set to the value of the **Option Explicit** setting in the **Options** dialog box. To view or change the setting in this dialog box, on the **Tools** menu, click **Options**. In the **Options** dialog box, expand **Projects and Solutions**, and then click **VB Defaults**. The initial default setting of **Option Explicit** in **VB Defaults** is **On**.

Setting **Option Explicit** to [off](#) is generally not a good practice. You could misspell a variable name in one or more locations, which would cause unexpected results when the program is run.

Option strict

Specifies whether to enforce strict type semantics. When **Option Strict** is **On**, the following conditions cause a compile-time error:

- Implicit narrowing conversions
- Late binding
- Implicit typing that results in an [Object](#) type

Implicit narrowing conversion errors occur when there is an implicit data type conversion that is a narrowing conversion. For more information, see [Option Strict Statement](#), [Implicit and Explicit Conversions](#), and [Widening and Narrowing Conversions](#).

An object is late bound when it is assigned to a property or method of a variable that is declared to be of type [Object](#). For more information, see [Option Strict Statement](#) and [Early and Late Binding](#).

Implicit object type errors occur when an appropriate type cannot be inferred for a declared variable, so a type of [Object](#) is inferred. This primarily occurs when you use a [Dim](#) statement to declare a variable without using an [As](#) clause, and [Option Infer](#) is off. For more information, see [Option Strict Statement](#), [Option Infer Statement](#), and the [Visual Basic Language Specification](#).

The **Option Strict** setting corresponds to the [/optionstrict](#) compiler option.

If a source code file contains an [Option Strict Statement](#), the [On](#) or [off](#) value in the statement overrides the **Option Strict** setting on the **Compile page**.

When you create a project, the **Option Strict** setting on the **Compile page** is set to the value of the **Option Strict** setting in the **Options** dialog box. To view or change the setting in this dialog box, on the **Tools** menu, click **Options**. In the **Options** dialog box, expand **Projects and Solutions**, and then click **VB Defaults**. The initial default setting of **Option Strict** in **VB Defaults** is **Off**.

Option Strict Individual Warnings

The **Warning configurations** section of the **Compile page** has settings that correspond to the three conditions that cause a compile-time error when [Option Strict](#) is on. Following are these settings:

- **Implicit conversion**
- **Late binding; call could fail at run time**
- **Implicit type; object assumed**

When you set **Option Strict** to **On**, all three of these warning configuration settings are set to **Error**. When you set **Option Strict** to **Off**, all three settings are set to **None**.

You can individually change each warning configuration setting to **None**, **Warning**, or **Error**. If all three warning configuration settings are set to **Error**, **On** appears in the **Option strict** box. If all three are set to **None**, **Off** appears in this box. For any other combination of these settings, **(custom)** appears.

Option compare

Specifies the type of string comparison to use. Select **Binary** to instruct the compiler to use binary, case-sensitive string comparisons. Select **Text** to use locale-specific, case-insensitive text string comparisons.

This setting corresponds to the [/optioncompare](#) compiler option.

If a source code file contains an [Option Compare Statement](#), the **Binary** or **Text** value in the statement overrides the **Option Compare** setting on the **Compile page**.

When you create a project, the **Option Compare** setting on the **Compile page** is set to the value of the **Option Compare** setting in the **Options** dialog box. To view or change the setting in this dialog box, on the **Tools** menu, click **Options**. In the **Options** dialog box, expand **Projects and Solutions**, and then click **VB Defaults**. The initial default setting of **Option Compare** in **VB Defaults** is **Binary**.

Option infer

Specifies whether to allow local type inference in variable declarations. Select **On** to allow the use of local type inference. Select **Off** to block local type inference.

This setting corresponds to the [/optioninfer](#) compiler option.

If a source code file contains an [Option Infer Statement](#), the **On** or **Off** value in the statement overrides the **Option Infer** setting on the **Compile page**.

When you create a project, the **Option Infer** setting on the **Compile page** is set to the value of the **Option Infer** setting in the **Options** dialog box. To view or change the setting in this dialog box, on the **Tools** menu, click **Options**. In the **Options** dialog box, expand **Projects and Solutions**, and then click **VB Defaults**. The initial default setting of **Option Infer** in **VB Defaults** is **On**.

Target CPU

Specifies the processor to be targeted by the output file. Specify **x86** for any 32-bit Intel-compatible processor, **x64** for any 64-bit Intel-compatible processor, **ARM** for any ARM processor, or **Any CPU** to specify that any processor is acceptable. **Any CPU** is the default value for new projects because it allows the application to run on the largest number of hardware types.

For more information, see [/platform \(Visual Basic\)](#).

Prefer 32-bit

If the **Prefer32-bit** check box is selected, the application runs as a 32-bit application on both 32-bit and 64-bit versions of Windows. Otherwise, the application runs as a 32-bit application on 32-bit versions of Windows and as a 64-bit application on 64-bit versions of Windows.

Running as a 64-bit application doubles the pointer size, and it can cause compatibility problems with libraries that are exclusively 32-bit. It makes sense to run an application as 64-bit only if it runs significantly faster or needs more than 4 GB of memory.

This check box is available only if all of the following conditions are true:

- On the **Compile Page**, the **Target CPU** list is set to **Any CPU**.
- On the **Application Page**, the **Application type** list specifies that the project is an application.
- On the **Application Page**, the **Target framework** list specifies the .NET Framework 4.5.

Warning configurations

This table lists build conditions and the corresponding notification level of **None**, **Warning**, or **Error** for each.

By default, all compiler warnings are added to the Task List during compilation. Select **Disable all warnings** to instruct the compiler not to issue warnings or errors. Select **Treat all warnings as errors** if you want the compiler to treat warnings as errors that must be fixed.

Disable all warnings

Specifies whether to allow the compiler to issue notifications as specified in the **Condition and Notification** table described earlier in this document. By default, this check box is cleared. Select this check box to instruct the compiler not to issue warnings or errors.

This setting corresponds to the [/nowarn](#) compiler option.

Treat all warnings as errors

Specifies how to treat warnings. By default, this check box is cleared, so that all warning notifications remain set to **Warning**. Select this check box to change all warning notifications to **Error**.

This option is available only if **Disable all warnings** is cleared.

Generate XML documentation file

Specifies whether to generate documentation information. By default, this check box is selected, instructing the compiler to generate documentation information and include it in an XML file. Clear this check box to instruct the compiler not to create documentation.

This setting corresponds to the [/doc](#) compiler option.

Register for COM interop

Specifies whether your managed application will expose a COM object (a COM-callable wrapper) that enables a COM object to interact with the application.

By default, this check box is cleared, which specifies that the application will not allow COM interop. Select this check box to allow COM interop.

This option is not available for Windows Application or Console Application projects.

Build Events

Click this button to access the **Build Events** dialog box. Use this dialog box to specify pre-build and post-build configuration instructions for the project. This dialog box applies to Visual Basic projects only. For more information, see [Build Events Dialog Box \(Visual Basic\)](#).

Advanced Compile Options

Click this button to access the **AdvancedCompiler Settings** dialog box. Use the **AdvancedCompiler Settings** dialog box to specify a project's advanced build configuration properties. This dialog box applies to Visual Basic projects only. For more information, see [Advanced Compiler Settings Dialog Box \(Visual Basic\)](#).

See also

- [How to: Specify Build Events \(Visual Basic\)](#)
- [Visual Basic Command-Line Compiler](#)
- [How to: Create and Edit Configurations](#)

Advanced Compiler Settings Dialog Box (Visual Basic)

8/9/2019 • 2 minutes to read • [Edit Online](#)

Use the **AdvancedCompiler Settings** dialog box of the **Project Designer** to specify the project's advanced build-configuration properties. This dialog box applies to Visual Basic projects only.

To access this dialog box

1. In **Solution Explorer**, choose a project node (not the **Solution** node).
2. On the **Project** menu, click **Properties**. When the **Project Designer** appears, click the **Compile** tab.
3. On the [Compile Page, Project Designer \(Visual Basic\)](#), select the **Configuration** and **Platform**. In simplified build configurations, the **Configuration** and **Platform** lists are not displayed. For more information, see [How to: Set debug and release configurations](#).
4. Click **Advanced Compile Options**.

NOTE

Your computer might show different names or locations for some of the Visual Studio user interface elements in this article. You may be using a different edition of Visual Studio or different environment settings. For more information, see [Personalize the IDE](#).

Optimizations

The following options specify optimizations that can in some cases make a program file smaller, make a program run faster, or speed up the build process.

Remove integer overflow checks

This check box is cleared, by default, to enable integer overflow checking. Select this check box to remove integer overflow checking. If you select this check box, integer calculations might be faster. However, if you remove overflow checking and data type capacities overflow, incorrect results might be stored without an error being raised.

If overflow conditions are checked and an integer operation overflows, an [OverflowException](#) exception is thrown. If overflow conditions are not checked, integer operation overflows don't throw an exception.

Enable optimizations

This check box is cleared, by default, to disable compiler optimizations. Select this check box to enable compiler optimizations. Compiler optimizations make your output file smaller, faster, and more efficient. However, because optimizations cause code rearrangement in the output file, compiler optimizations can make debugging difficult.

DLL base address

This text box displays the default DLL base address in hexadecimal format. In Class Library and Control Library projects, you can use this text box to specify the base address to be used when the DLL is created.

Generate debug info

Select **None**, **Full**, or **pdb-only** from the list. **None** specifies that no debugging information be generated. **Full** specifies that full debugging information be generated, and **pdb-only** specifies that only PDB debugging

information should be generated. The default value for this option is **Full**.

Compilation Constants

Conditional compilation constants have an effect similar to that of using a `#Const` preprocessor directive in a source file, except that constants defined are public and apply to all files in the project. You can use conditional compilation constants together with the `#If...Then...#Else` directive to compile source files conditionally. See [Conditional Compilation](#).

Define DEBUG constant

By default, this check box is selected, specifying that a DEBUG constant be set.

Define TRACE constant

By default, this check box is selected, specifying that a TRACE constant be set.

Custom constants

Enter any custom constants for your application in this text box. Entries should be delimited by commas, using this form: `Name1="Value1",Name2="Value2",Name3="Value3"`.

Other Settings

Generate serialization assemblies

This setting specifies whether the compiler will create XML serialization assemblies. Serialization assemblies can improve the startup performance of `XmlSerializer` if you've used that class to serialize types in your code. The default value for this option is **Auto**. **Auto** specifies that serialization assemblies be generated only if you've used `XmlSerializer` to encode types in your code to XML. **Off** specifies that serialization assemblies never be generated, regardless of whether your code uses `XmlSerializer`. **On** specifies that serialization assemblies always be generated. Serialization assemblies are named `TypeName.XmlSerializers.dll`.

See also

- [Compile Page, Project Designer \(Visual Basic\)](#)

Build Events Dialog Box (Visual Basic)

7/24/2019 • 2 minutes to read • [Edit Online](#)

Use the **Build Events** dialog box to specify build configuration instructions. You can also specify the conditions under which any pre-build or post-build events are run. For more information, see [How to: Specify Build Events \(Visual Basic\)](#).

Pre-build event command line

Specifies any commands to execute before the build starts. To type long commands, click **Edit Pre-build** to display the [Pre-build Event/Post-build Event Command Line Dialog Box](#).

NOTE

Pre-build events do not run if the project is up-to-date and no build is triggered.

Post-build event command line

Specifies any commands to execute after the build ends. To type long commands, click **Edit Post-build** to display the [Pre-build Event/Post-build Event Command Line](#) dialog box.

NOTE

Add a `call` statement before all post-build commands that run .bat files. For example, `call C:\MyFile.bat` or
`call C:\MyFile.bat call C:\MyFile2.bat`.

Run the post-build event

Specifies the conditions for the post-build event to run, as shown in the following table.

OPTION	RESULT
Always	Post-build event will run, regardless of whether the build succeeds.
On successful build	Post-build event will run if the build succeeds. The event will run even for a project that is up-to-date, as long as the build succeeds. This is the default setting.
When the build updates the project output	Post-build event will run only when the compiler's output file (.exe or .dll) differs from the previous compiler output file. A post-build event is not run if a project is up-to-date.

See also

- [Compile Page, Project Designer \(Visual Basic\)](#)
- [How to: Specify Build Events \(Visual Basic\)](#)
- [Pre-build Event/Post-build Event Command Line Dialog Box](#)

Debug Page, Project Designer

7/24/2019 • 2 minutes to read • [Edit Online](#)

Use the **Debug** page of the **Project Designer** to set properties for debugging behavior in a Visual Basic or C# project.

To access the **Debug** page, select a project node in **Solution Explorer**. On the **Project** menu, choose **<ProjectName> Properties**. When the **Project Designer** appears, click the **Debug** tab.

NOTE

This topic does not apply to UWP apps. See [Start a debug session \(VB, C#, C++ and XAML\)](#) for UWP apps.

Configuration and Platform

The following options allow you to select the configuration and platform to display or modify.

Configuration

Specifies which configuration settings to display or modify. The settings can be **Debug** (default), **Release**, or **All Configurations**.

Platform

Specifies which platform settings to display or modify. The choices can include **Any CPU** (default), **x64**, and **x86**.

Start action

Start action indicates the item to start when the application is debugged: the project, a custom program, a URL, or nothing. By default, this option is set to **Start project**. The **Start Action** setting on the **Debug** page determines the value of the `StartAction` property.

Start project

Choose this option to specify that the executable (for Windows Application and Console Application projects) should be started when the application is debugged. This option is selected by default.

Start external program

Choose this option to specify that a specific program should be started when the application is debugged.

Start browser with URL

Choose this option to specify that a particular URL should be accessed when the application is debugged.

Start options

Command line arguments

In this text box, enter the command-line arguments to use for debugging.

Working directory

In this text box, enter the directory from which the project will be launched. Or click the Browse button (...) to choose a directory.

Use remote machine

To debug the application from a remote computer, select this check box, and enter the path to the remote computer in the text box.

Debugger engines

Enable native code debugging

This option specifies whether debugging of native code is supported. Select this check box if you are making calls to COM objects or if you start a custom program written in native code that calls your project and you must debug the native code. Clear this check box to disable debugging of unmanaged code. This check box is cleared by default.

Enable SQL Server debugging

Select or clear this check box to enable or disable debugging of SQL procedures from your Visual Basic application. This check box is cleared by default.

See also

- [First look at the debugger](#)
- [Project Settings for C# Debug Configurations](#)
- [Project Settings for a Visual Basic Debug Configuration](#)
- [How to: Debug a ClickOnce Application with Restricted Permissions](#)
- [How to: Create and Edit Configurations](#)

My Extensions Page, Project Designer (Visual Basic)

10/21/2019 • 2 minutes to read • [Edit Online](#)

Use the **My Extensions** page of the **Project Designer** to manage `My` namespace extensions in your project. `My` namespace extensions enable you to customize the `My` namespace to add your own custom members. For information about creating custom `My` namespace extensions, see [Extending the My Namespace in Visual Basic](#).

To access the **My Extensions** page, double-click **My Project** for your project node in **Solution Explorer**. When the **Project Designer** appears, click the **My Extensions** tab.

UIElement List

The following options enable you to add or remove `My` namespace extensions in your project. A `My` namespace extension must first be installed as a Visual Studio Item template to be available to be added. For information about publishing and installing `My` namespace extensions, see [Packaging and Deploying Custom My Extensions](#).

My namespace extensions

This list shows all the `My` namespace extensions installed in the project.

Add Extension

Click this button to add an installed `My` namespace extension to your project. A list of all possible `My` namespace extensions will appear. Select the `My` namespace extension that you want to add to your project and click **OK** to add it.

Remove Extension

Select one or more references in the **My namespace extensions** list, and then click this button to remove the `My` namespace extension from the project.

See also

- [Extending the My Namespace in Visual Basic](#)
- [Packaging and Deploying Custom My Extensions](#)
- [Extending the Visual Basic Application Model](#)
- [Customizing Which Objects are Available in My](#)

Publish Page, Project Designer

8/9/2019 • 3 minutes to read • [Edit Online](#)

The **Publish** page of the **Project Designer** is used to configure properties for ClickOnce deployment.

To access the **Publish** page, select a project node in **Solution Explorer**, and then, on the **Project** menu, click **Properties**. When the **Project Designer** appears, click the **Publish** tab.

NOTE

Some of the ClickOnce properties described here can also be set in the **PublishWizard**, available from the **Build** menu or by clicking the **PublishWizard** button on this page.

UIElement List

Publishing Folder Location

Specifies the location where the application is published. Can be a drive path (`C:\deploy\myapplication`), a file share (`\server\myapplication`), or an FTP server (`ftp://ftp.microsoft.com/myapplication`). Note that text must be present in the **Publishing Location** box in order for the browse (...) button to work.

Installation Folder URL

Optional. Specifies a website to which users go to install the application. This is necessary only when it differs from the **Publishing Location**, for example, when the application is published to a staging server.

Install Mode and Settings

Determines whether the application is run directly from the **Publishing Location** (when **The application is available online only** is selected) or is installed and added to the **Start** menu and the **Add or Remove Programs** item in **Control Panel** (when **The application is available offline as well** is selected).

For WPF web browser apps, the **The application is available offline as well** option is disabled, because such applications are available only online.

Application Files

Opens the Application Files dialog box, which is used to specify how and where individual files are installed.

Prerequisites

Opens the Prerequisites dialog box, which is used to specify prerequisite components, such as the .NET Framework, to be installed together with the application.

Updates

Opens the Application Updates dialog box, which is used to specify update behavior for the application. Not available when **The application is available online only** is selected.

Options

Opens the Publish Options dialog box, which is used to specify additional advanced publishing options.

Publish Version

Sets the publish version number for the application; when the version number is changed, the application is

published as an update. Each part of the publish version (**Major**, **Minor**, **Build**, **Revision**) can have a maximum value of 65355 ([MaxValue](#)), the maximum allowed by [Version](#).

When you install more than one version of an application by using ClickOnce, the installation moves earlier versions of the application into a folder named Archive, in the publish location that you specify. Archiving earlier versions in this manner keeps the installation directory clear of folders from the earlier version.

Automatically increment revision with each publish

Optional. When this option is selected (the default), the **Revision** part of the publish version number is incremented by one every time that the application is published. This causes the application to be published as an update.

Publish Wizard

Opens the Publish Wizard. Completing the Publish Wizard has the same effect as running the **Publish** command on the **Build** menu.

Publish Now

Publishes the application using the current settings. Equivalent to the **Finish** button in the **Publish Wizard**.

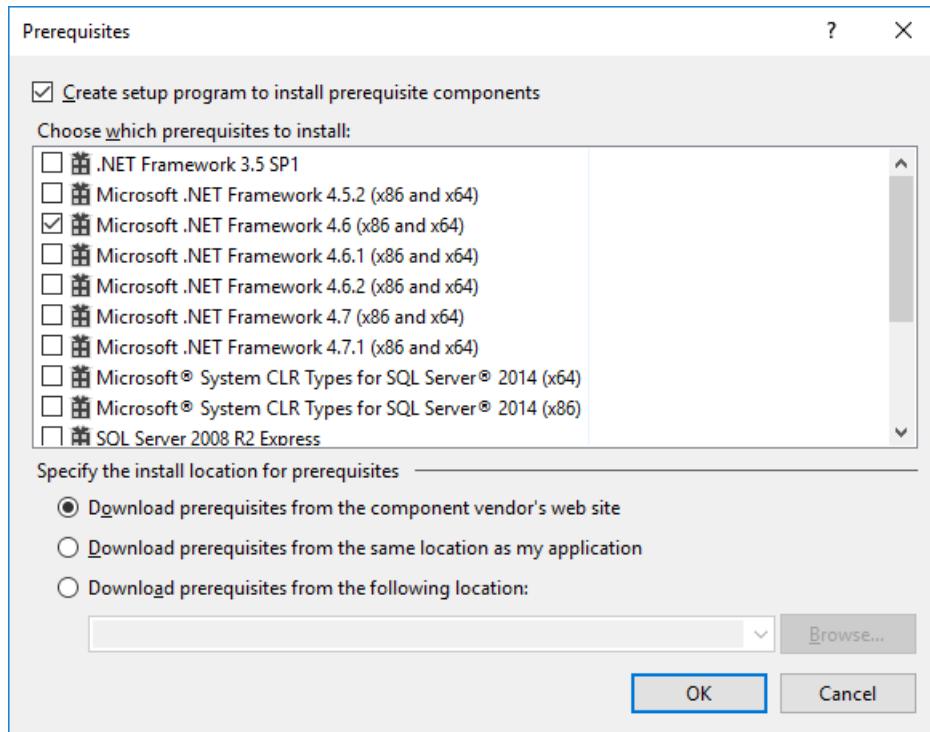
See also

- [Publishing ClickOnce Applications](#)
- [How to: Publish a ClickOnce Application using the Publish Wizard](#)
- [How to: Specify Where Visual Studio Copies the Files](#)
- [How to: Specify the Location Where End Users Will Install From](#)
- [How to: Specify a Link for Technical Support](#)
- [How to: Specify the ClickOnce Offline or Online Install Mode](#)
- [How to: Enable AutoStart for CD Installations](#)
- [How to: Set the ClickOnce Publish Version](#)
- [How to: Automatically Increment the ClickOnce Publish Version](#)
- [How to: Specify Which Files Are Published by ClickOnce](#)
- [How to: Install Prerequisites with a ClickOnce Application](#)
- [How to: Manage Updates for a ClickOnce Application](#)
- [How to: Change the Publish Language for a ClickOnce Application](#)
- [How to: Specify a Start Menu Name for a ClickOnce Application](#)
- [How to: Specify a Publish Page for a ClickOnce Application](#)
- [ClickOnce Security and Deployment](#)

Prerequisites dialog box

7/26/2019 • 3 minutes to read • [Edit Online](#)

The **Prerequisites** dialog box specifies which prerequisite components are installed, how they are installed, and which order the packages are installed.



To access the dialog box, select a project node in **Solution Explorer**, and then select **Project > Properties**. When the **Project Designer** appears, select the **Publish** tab, and then select **Prerequisites**. For Setup projects, on the **Project** menu, click **Properties**. When the **Property Pages** dialog box appears, click **Prerequisites**.

UIElement list

ELEMENT	DESCRIPTION
Create setup program to install prerequisite components	Includes the prerequisite components in your application's Setup program (<i>Setup.exe</i>) so that they'll be installed before your application, in order of dependency. By default, this option is selected. If it is not selected, no <i>Setup.exe</i> is created.
Choose which prerequisites to install	Specifies whether to install components such as .NET Framework and C++ runtime libraries. For example, by selecting the check box next to SQL Server 2012 Express , you specify that the Setup program must verify whether this component is installed on the target computer, and install it if it's not. For detailed information about each prerequisite package, see Prerequisites information .
Download prerequisites from the component vendor's web site	Specifies that the prerequisite components be installed from the vendor's website. This is the default option.

ELEMENT	DESCRIPTION
Download prerequisites from the same location as my application	Specifies that the prerequisite components be installed from the same location as the application. This copies all the prerequisite packages to the publish location. For this option to work, the prerequisite packages must be on the development computer.
Download prerequisites from the following location	Specifies that the prerequisite components be installed from the location that you enter. You can use the Browse button to select a location.

NOTE

For information on where to put prerequisites, see [Create bootstrapper packages](#).

Prerequisites information

The prerequisite components that appear in the **Prerequisites** dialog box might differ from those in the following list. The prerequisite packages listed in the **Prerequisites Dialog Box** are set automatically the first time that you open the dialog box. If you subsequently change the project's target framework, you have to select the prerequisites manually to match the new target framework.

ELEMENT	DESCRIPTION
.NET Framework 3.5 SP1	<p>This package installs the following:</p> <ul style="list-style-type: none"> - .NET Framework versions 2.0, 3.0, and 3.5. - Support for all .NET Framework versions on 32-bit (x86) and 64-bit (x64) operating systems. - Language packs for each .NET Framework version that is installed with the package. - Service packs for .NET Framework 2.0 and 3.0. <p>.NET Framework 3.0 is included with Windows Vista, and .NET Framework 3.5 is included with Visual Studio. .NET Framework 3.5 is required for all Visual Basic and C# projects that are compiled for 32-bit operating systems and for which the target framework is set to .NET Framework 3.5, and for Visual Basic and C# projects compiled for 64-bit operating systems. (IA64 is not supported.) Note that Visual Basic and C# projects are compiled for any CPU architecture by default. For more information, see Framework targeting overview and Deploy prerequisites for 64-bit apps.</p>
Microsoft .NET Framework 4.x	This package installs the .NET Framework 4.x for both the x86 and x64 platforms.
Microsoft System CLR Types for SQL Server 2014 (x64 and x86)	This package installs Microsoft System CLR Types for SQL Server 2014 for either x64 or x86.
SQL Server 2008 R2 Express	This package installs Microsoft SQL Server 2008 R2 Express, a free edition of Microsoft SQL Server 2008 R2, an ideal database for small web, server, or desktop applications. It can be used for free for development and production.
SQL Server 2012 Express	This package installs Microsoft SQL Server 2012 Express.

ELEMENT	DESCRIPTION
SQL Server 2012 Express LocalDB	This package installs Microsoft SQL Server 2012 Express LocalDB.
Visual C++ "14" Runtime Libraries (ARM)	<p>This package installs the Visual C++ run-time libraries for the Itanium architecture, which provide routines for programming for the Microsoft Windows operating system. These routines automate many common programming tasks that are not provided by the C and C++ languages.</p> <p>For more information, see C Run-Time Library Reference.</p>
Visual C++ "14" Runtime Libraries (x64)	<p>This package installs the Visual C++ run-time libraries for the x64 operating systems, which provide routines for programming for the Microsoft Windows operating system. These routines automate many common programming tasks that are not provided by the C and C++ languages.</p> <p>For more information, see C Run-Time Library Reference.</p>
Visual C++ "14" Runtime Libraries (x86)	<p>This package installs the Visual C++ run-time libraries for the x86 operating systems, which provide routines for programming for the Microsoft Windows operating system. These routines automate many common programming tasks that are not provided by the C and C++ languages.</p> <p>For more information, see C Run-Time Library Reference.</p>

See also

- [Publish Page, Project Designer](#)
- [Application Deployment Prerequisites](#)
- [Deploying Prerequisites for 64-bit Applications](#)
- [Framework targeting overview](#)

References Page, Project Designer (Visual Basic)

10/21/2019 • 2 minutes to read • [Edit Online](#)

Use the **References** page of the **Project Designer** to manage references, web references, and imported namespaces in your project. Projects can contain references to COM components, XML web services, .NET libraries or assemblies, or other class libraries. For more information on using references, see [Managing references in a project](#).

To access the **References** page, choose a project node (not the **Solution** node) in **Solution Explorer**. Then choose **Project, Properties** on the menu bar. When the Project Designer appears, click the **References** tab.

UIElement List

The following options allow you to select or remove references and imported namespaces in your project.

Reference Paths

Click this button to access the **Reference Paths** dialog box.

NOTE

When the project system finds an assembly reference, the system resolves the reference by looking in the following locations, in the following order:

1. The project folder. The project folder files appear in **Solution Explorer** when **Show All Files** isn't in effect.
2. Folders that are specified in the **Reference Paths** dialog box.
3. Folders that display files in the **Add Reference** dialog box.
4. The project's obj folder. (When you add a COM reference to your project, one or more assemblies may be added to the project's obj folder.)

References

This list shows all references in the project, used or unused.

Add

Click this button to add a reference or web reference to the **References** list.

Choose **Reference** to add a reference to your project using the Add Reference dialog box.

Choose **Web Reference** to add a web reference to your project using the **Add Web Reference** dialog box.

Remove

Select one or more references in the **References** list, then click this button to delete it.

Update Web Reference

Select a web reference in the **References** list and click this button to update it.

Imported namespaces

You can type your own namespace in this box and click **Add User Import** to add it to the namespaces list.

You can create aliases for user-imported namespaces. To do this, enter the alias and the namespace in the format *alias=namespace*. This is useful if you are using long namespaces, for example:

```
Http= MyOrg.ObjectLib.Internet.WebRequestMethods.Http .
```

Add User Import

Click this button to add the namespace specified in the **Imported namespaces** box to the list of imported namespaces. The button is active only if the specified namespace is not already in the list.

Namespaces list

This list shows all available namespaces. The check boxes for namespaces included in your project are selected.

Update User Import

Select a user-specified namespace in the namespaces list, type the name that you want to replace it with in the **Imported namespaces** box, and then click this button to change to the new namespace. The button is active only if the selected namespace is one that you added to the list by using the **Add User Import** button. You can add:

- Classes or namespaces, such as `System.Math`.
- Aliased imports, such as `VB=Microsoft.VisualBasic`.
- XML namespaces, such as `<xmlns:xsl="http://www.w3.org/1999/XSL/Transform">`.

See also

- [Managing references in a project](#)
- [How to: Add or Remove Imported Namespaces \(Visual Basic\)](#)
- [Imports Statement \(XML Namespace\)](#)

Security Page, Project Designer

10/21/2019 • 2 minutes to read • [Edit Online](#)

The **Security** page of the **Project Designer** is used to configure code access security settings for applications that are deployed by using ClickOnce deployment. For more information, see [Code Access Security for ClickOnce Applications](#).

To access the **Security** page, click a project node in **Solution Explorer**, and then, on the **Project** menu, click **Properties**. When the **Project Designer** appears, click the **Security** tab.

Security Settings

Enable ClickOnce Security Settings

Determines whether security settings are enabled at design time. When this option is cleared, all other options on the **Security** page are unavailable.

NOTE

When you publish an application by using the **Publish** wizard, this option is automatically enabled.

When you select this option, you have the choice of selecting one of two radio buttons: **This is a full trust application** or **This is a partial trust application**.

By default, for WPF Web Browser Application projects, this option is selected.

By default, for all other project types, this option is cleared.

This is a full trust application

If you select this option, the application requests Full Trust permissions when it is installed or run on a client computer. Avoid using Full Trust if possible, because your application will be granted unrestricted access to resources such as the file system and the registry.

By default, for WPF Web Browser Application projects, this option is set to Partial Trust.

By default, for all other project types, this option is set to Full Trust.

This is a partial trust application

If you select this option, the application requests Partial Trust permissions when it is installed or run on a client computer. *Partial Trust* means that only the actions that are permitted under the requested code access security permissions will run. For more information about how to configure security permissions, see [Code Access Security for ClickOnce Applications](#).

You can specify the Partial Trust security settings by configuring the options in the **ClickOnce Security Permissions** area.

ClickOnce Security Permissions

Zone your application will be installed from

Specifies a default set of code-access security permissions. Choose **Internet** or **Local Intranet** for a restricted permission set, or choose **(Custom)** to configure a custom permission set. If the application requests more

permissions than granted in a zone, a ClickOnce trust prompt appears for the end user to grant the additional permissions. For more information about how to configure security permissions, see [Code Access Security for ClickOnce Applications](#).

By default, for WPF Web Browser Application projects, this option is set to **Internet**.

Edit Permissions XML

Opens the application manifest template (app.manifest) to configure the permissions for the **(Custom)** permission set.

Advanced

Opens the [Advanced Security Settings Dialog Box](#), which is used to configure settings for debugging the application with restricted permissions. These settings are checked during debugging, and permission exceptions indicate that your application may need more permissions than defined in a zone.

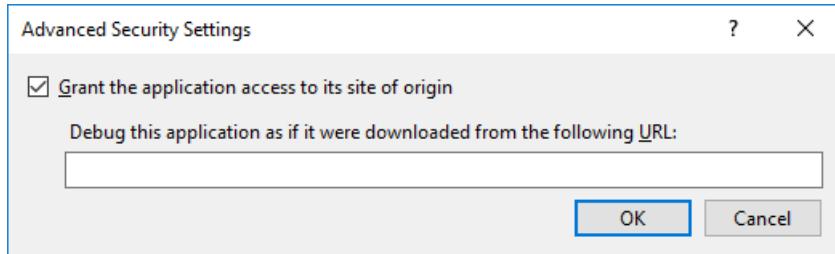
See also

- [WebBrowserPermission](#)
- [MediaPermission](#)
- [Code Access Security for ClickOnce Applications](#)
- [How to: Enable ClickOnce Security Settings](#)
- [How to: Set a Security Zone for a ClickOnce Application](#)
- [How to: Set Custom Permissions for a ClickOnce Application](#)
- [How to: Debug a ClickOnce Application with Restricted Permissions](#)
- [ClickOnce Security and Deployment](#)
- [Project Properties Reference](#)
- [Advanced Security Settings Dialog Box](#)

Advanced Security Settings dialog box

7/24/2019 • 2 minutes to read • [Edit Online](#)

This dialog box allows you to specify security settings related to debugging in zone.



To access this dialog box, select a project node in **Solution Explorer**, and then, on the **Project** menu, click **Properties**. When the **Project Designer** appears, click the **Security** tab. On the **Security** page, select **Enable ClickOnce Security Settings**, click **This is a partial trust application**, and then click **Advanced**.

UIElement list

Grant the application access to its site of origin

If you select this check box, the application can access the website or server share on which it is published. By default, this option is selected.

Debug this application as if it were downloaded from the following URL

If you have to allow the application to access the website or server share corresponding to the **Installation URL** you specified on the **Publish** page, enter that URL here. This option is available only when **Grant the application access to its site of origin** is selected.

See also

- [Security Page, Project Designer](#)

Services Page, Project Designer

10/18/2019 • 2 minutes to read • [Edit Online](#)

Client application services provide simplified access to Microsoft Ajax login, roles, and profile services from Windows Forms and Windows Presentation Foundation (WPF) applications. You can use the **Services** page of the **Project Designer** to enable and configure client application services for your project.

With client application services, you can use a centralized server to authenticate users, determine each user's assigned role or roles, and store per-user application settings that you can share across the network. For more information, see [Client Application Services](#).

To access the **Services** page, select a project node in **Solution Explorer**, and then click **Properties** on the **Project** menu. When the **Project Designer** appears, click the **Services** tab.

Task List

[How to: Configure Client Application Services](#)

UIElement List

Configuration

This control is not editable on this page. For a description of this control, see [Compile Page, Project Designer \(Visual Basic\)](#) or [Build Page, Project Designer \(C#\)](#).

Platform

This control is not editable on this page. For a description of this control, see [Compile Page, Project Designer \(Visual Basic\)](#) or [Build Page, Project Designer \(C#\)](#).

Enable client application services

Select to enable client application services. You must specify service locations on the **Services** page to use client application services.

Use Windows authentication

Indicates that the authentication provider will use Windows-based authentication, that is, the identity supplied by the Windows operating system.

Use Forms authentication

Indicates that the authentication provider will use forms authentication. This means that your application must provide a user interface for login. For more information, see [How to: Implement User Login with Client Application Services](#).

Authentication service location

Used only with forms authentication. Specifies the location of the authentication service.

Optional: Credentials provider

Used only with forms authentication. Indicates the `IClientFormsAuthenticationCredentialsProvider` implementation that the authentication service will use to display a login dialog box when your application calls the `static System.Web.Security.Membership.ValidateUser` method and passes empty strings or `null` for the

parameters. If you leave this box blank, you must pass a valid user name and password to the [System.Web.Security.Membership.ValidateUser](#) method. You must specify the credentials provider as an assembly-qualified type name. For more information, see [System.Type.AssemblyQualifiedName](#) and [Assembly Names](#). In its simplest form, an assembly-qualified type name looks similar to the following example:

```
MyNamespace.MyLoginClass, MyAssembly
```

Roles service location

Specifies the location of the roles service.

Web settings service location

Specifies the location of the profile (web settings) service.

Advanced

Opens the [Advanced Settings for Services Dialog Box](#), which you can use to override default behavior. For example, you can use this dialog box to specify a database for offline storage instead of using the local file system. For more information, see [Advanced Settings for Services Dialog Box](#).

See also

- [Client Application Services](#)
- [Advanced Settings for Services Dialog Box](#)
- [How to: Configure Client Application Services](#)
- [Compile Page, Project Designer \(Visual Basic\)](#)
- [Build Page, Project Designer \(C#\)](#)

Advanced Settings for Services Dialog Box

10/18/2019 • 2 minutes to read • [Edit Online](#)

Client application services provide simplified access to Microsoft Ajax login, roles, and profile services from Windows Forms and Windows Presentation Foundation (WPF) applications. You can use the **Services** page in the **Project Designer** to configure client application services. For more information about the **Services** page, see [Services Page, Project Designer](#).

Use the **Advanced Settings for Services** dialog box of the **Services** page in the **Project Designer** to configure advanced settings for client application services. By using these settings, you can override some default application service behaviors to enable less common scenarios. For more information, see [Client Application Services](#).

To access the **Advanced Settings for Services** dialog box, select a project node in **Solution Explorer**, and then click **Properties** on the **Project** menu. When the **Project Designer** appears, click the **Services** tab, and then click the **Advanced** button. This button will be disabled until you enable client application services.

Task List

- [How to: Configure Client Application Services](#)

UIElement List

Save password hash locally to enable offline login Specifies whether an encrypted form of the user's password will be cached locally to enable the user to log in when the application is in offline mode. This option is selected by default.

Require users to log on again whenever the server cookie expires Specifies whether previously authenticated users are automatically reauthenticated when your application accesses the roles or profile service and the server authentication cookie has expired. Select this option to deny access to the application services and require explicit reauthentication after the cookie expires. This is useful for applications deployed in public locations to make sure that users who leave the application running after use will not remain authenticated indefinitely. This option is cleared by default.

Role service cache timeout Specifies the amount of time the client role provider will use cached role values instead of accessing the roles service. Set this time interval to a small value when roles are updated frequently or to a larger value when roles are updated infrequently. The default value is one day.

The role provider accesses the cached role values or the roles service when you call the `IsInRole` method. To programmatically clear the cache and force this method to access the remote service, call the `ResetCache` method.

Use custom connection string Specifies whether the client service providers will use a custom data store for the local cache. By default, the service providers will use the local file system for the cache. Selecting this option will automatically populate the text box with a default connection string. You can keep the default connection string to automatically generate and use a SQL Server Compact Edition database, or you can specify a connection string to an existing SQL Server database. For more information, see [How to: Configure Client Application Services](#). This option is cleared by default.

See also

- [Client Application Services](#)

- [Services Page, Project Designer](#)
- [How to: Configure Client Application Services](#)

Settings page, Project Designer

10/18/2019 • 3 minutes to read • [Edit Online](#)

Use the **Settings** page of the Project Designer to specify a project's application settings. Application settings enable you to store and retrieve property settings and other information for your application dynamically. They also enable you to maintain custom application and user preferences on a client computer. For more information, see [Manage application settings](#).

To access the **Settings** page, select a project node in **Solution Explorer**, and then select **Project > Properties**. When the Project Designer appears, select the **Settings** tab.

Header bar

The header bar at the top of the **Settings** page contains several controls:

Synchronize

Synchronize restores user-scoped settings that the application uses at run time or during debugging to their default values as defined at design time. To restore the data, remove run-time generated application-specific files from disk, not from project data.

Load Web Settings

Load Web Settings displays a **Login** dialog box that enables you to load settings for an authenticated user or for anonymous users. This button is enabled only when you've enabled client application services on the **Services** page and specified a **Web settings service location**.

View Code

For C# projects, the **View Code** button enables you to view the code in the *Settings.cs* file. This file defines the `Properties.Settings` class, which enables you to handle specific events on the `Settings` object. In languages other than Visual Basic, you must explicitly call the `Save` method of this wrapper class in order to persist the user settings. You usually do this in the **Closing** event handler of the main form. Following is an example of a call to the `Save` method:

```
Properties.Settings.Default.Save();
```

For Visual Basic projects, the **View Code** button enables you to view the code in the *Settings.vb* file. This file defines the `MySettings` class, which enables you to handle specific events on the `My.Settings` object. For more information about accessing application settings by using the `My.Settings` object, see [Access application settings](#).

For more information about accessing application settings, see [Application settings for Windows Forms](#).

Access modifier

The **Access modifier** button specifies the access level of the `Properties.Settings` (in C#) or `My.Settings` (in Visual Basic) helper classes that Visual Studio generates in *Settings.Designer.cs* or *Settings.Designer.vb*.

For Visual C# projects, the access modifier can be **Internal** or **Public**.

For Visual Basic projects, the access modifier can be **Friend** or **Public**.

By default, the setting is **Internal** in C# and **Friend** in Visual Basic. When Visual Studio generates helper classes as **Internal** or **Friend**, executable (.exe) applications cannot access the resources and settings that you have added to

class libraries (.dll files). If you have to share resources and settings from a class library, set the access modifier to **Public**.

For more information about the settings helper classes, see [Manage application settings](#).

Settings grid

Settings Grid is used to configure application settings. This grid includes the following columns:

Name

Enter the name of the application setting in this field.

Type

Use the drop-down list to select a type for the setting. The most frequently used types appear in the drop-down list, for example, **String**, **(Connection string)**, and **System.Drawing.Font**. You can choose another type by selecting **Browse** at the end of the list, and then selecting a type from the **Select a Type** dialog box. After you choose a type, it's added to the common types in the drop-down list (for the current solution only).

Scope

Select either **Application** or **User**.

Application-scoped settings, such as connection strings, are associated with the application. Users can't change application-scoped settings at run time.

User-scoped settings, such as system fonts, are intended to be used for user preferences. Users can change them at run time.

Value

The data or value associated with the application setting. For example, if the setting is a font, its value could be **Verdana, 9.75pt, style=Bold**.

See also

- [Manage application settings](#)
- [Access application settings \(Visual Basic\)](#)

Signing Page, Project Designer

7/24/2019 • 4 minutes to read • [Edit Online](#)

Use the **Signing** page of the **Project Designer** to sign the application and deployment manifests and also to sign the assembly (strong-name signing).

Notice that the signing of application and deployment manifests is a process distinct from the signing of an assembly, although both tasks are performed on the **Signing** page.

Also, the storage of key-file information differs for manifest signing and assembly signing. For manifest signing, key information is stored in your computer's cryptographic storage database and the current user's Windows certificate store. For assembly signing, key information is stored only in your computer's cryptographic storage database.

To access the **Signing** page, select a project node in **Solution Explorer**, and then, on the **Project** menu, click **Properties**. When the **Project Designer** appears, click the **Signing** tab.

Application and Deployment Manifest Signing

Sign the ClickOnce manifests check box

Select this check box to sign the application and deployment manifests with a public/private key pair. For more information about how to do this, see [How to: Sign Application and Deployment Manifests](#).

Select from Store button

Allows you to select an existing certificate from the current user's personal certificate store. You can select one of these certificates to sign your application and deployment manifests.

Clicking **Select from Store** opens the **Select a Certificate** dialog box, which lists certificates in your personal certificate store that are currently valid (not expired) and that have private keys. The purpose of the certificate you select should include code signing.

If you click **view certificate properties**, the **Certificate Details** dialog box appears. This dialog box includes detailed information about the certificate, and includes additional options. You can click **Learn more about certificates** to view additional Help information.

Select from File button

Allows you to select a certificate from an existing key file.

Clicking **Select from File** opens the **Select File** dialog box, which enables you to select a certificate key (.pfx) file. The file must be password protected and cannot already be located in your personal certificate store.

In the **Enter password to open file** dialog box, enter a password to open the certificate key (.pfx) file. The password information is stored in your personal key container list and your personal certificate store.

Create Test Certificate button

Allows you to create a certificate for testing. The test certificate is used for signing your ClickOnce application and deployment manifests.

Clicking **Create Test Certificate** opens the **Create Test Certificate** dialog box, in which you can type a password for the strong-name key file for the test certificate. The file is named *projectname_TemporaryKey.pfx*. If you click **OK** without typing a password, the .pfx file is not password encrypted.

Timestamp server URL box

Specifies the address of a server that timestamps your signature. When you provide a certificate, this external site verifies the time that the application was signed.

Assembly Signing

Sign the assembly check box

Select this check box to sign the assembly and create a strongly named key file. For more information about signing the assembly by using the **Project Designer**, see [How to: Sign an Assembly \(Visual Studio\)](#).

This option uses the Al.exe tool provided by the Windows Software Development Kit (SDK) to sign the assembly. For more information about Al.exe, see [How to: Sign an Assembly with a Strong Name](#).

Choose a strong name key file list

Enables you to specify a new or existing strongly named key file that is used to sign the assembly. Select <**Browse...**> to select an existing key file.

Select <**New...**> to create a new key file with which to sign the assembly. The **Create Strong Name Key** dialog box appears, which you can use to specify a key file name and protect the key file with a password. The password must be at least 6 characters long. If you specify a password, a Personal Information Exchange (.pfx) file is created; if you do not specify a password, a strongly named key (.snk) file is created.

Change Password button

Changes the password for the Personal Information Exchange (.pfx) key file that is used to sign the assembly.

Clicking **Change Password** opens the **Change Key Password** dialog box. In this dialog box, **Old password** is the current password for the key file. **New password** must be at least 6 characters long. The password information is stored in current user's Windows certificate store.

Delay sign only check box

Select this check box to enable delay signing.

Note that a delay signed project will not run and cannot be debugged. You can, however, use the [Sn.exe \(Strong Name Tool\)](#) with the `-vr` option to skip verification during development.

NOTE

When you sign an assembly, you might not always have access to a private key. For example, an organization might have a closely guarded key pair that developers don't have access to on a daily basis. The public key might be available, but access to the private key is restricted to a few individuals. In such a case, you can use *delayed* or *partial signing* to provide the public key, deferring the addition of the private key until the assembly is handed off.

See also

- [Project Properties Reference](#)
- [Managing Assembly and Manifest Signing](#)
- [How to: Sign Application and Deployment Manifests](#)
- [How to: Sign an Assembly \(Visual Studio\)](#)
- [How to: Sign an Assembly with a Strong Name](#)
- [Strong-Named Assemblies](#)

Property pages, JavaScript

8/9/2019 • 3 minutes to read • [Edit Online](#)

The **Property Pages** provides access to project settings. You can use the pages that appear in the **Property Pages** to change project properties.

To access the project properties, select a project node in **Solution Explorer**. On the **Project** menu, click **Properties**.

NOTE

Your computer might show different names or locations for some of the Visual Studio user interface elements in this article. You may be using a different edition of Visual Studio or different environment settings. For more information, see [Personalize the IDE](#).

The following pages and options appear in the **Property Pages**.

Configuration and Platform Page

Use the following options to select the configuration and platform to display or modify.

Configuration

Specifies the configuration settings to display or modify. The settings are **Debug** (default), **Release**, **All Configurations**, or a user-defined configuration. For more information, see [How to: Set debug and release configurations in Visual Studio](#).

Platform

Specifies the platform settings to display or modify. The settings are **Any CPU** (default for Windows 8.x Store apps), **x64**, **ARM**, **x86**, or a user-defined platform. For more information, see [How to: Set debug and release configurations in Visual Studio](#).

General Page

Use the following options to set general properties of the project.

NOTE

Some options are only available in UWP apps.

Output Path

Specifies the location of the output files for the project's configuration. The path is relative; if you enter an absolute path, the absolute path is saved in the project. The default path is bin\Debug.

When you use simplified build configurations, the project system determines whether to build a debug or release version. When you click **Debug > Start Debugging** (or press **F5**), the build is put in the debug location regardless of the **Output path** you specify. However, the **Build Solution** command on the **Build** menu puts it in the location you specify. To enable advanced build configurations, on the menu bar, choose **Tools > Options**. In the **Options** dialog box, expand **Projects and Solutions**, select **General**, and then clear the **Show advanced build configurations** check box. This gives you manual control over all configuration values and whether a debug or

release version is built.

Default Language

Specifies the default language for the project. The language option selected in **Clock, Language, and Region** in Control Panel specifies the user's preferred language. By specifying a default language for the project, you make sure that the specified default language resources are used if the user's preferred language does not match the language resources provided in the application.

Debug Page

Use the following options to set properties for debugging behavior in the project.

NOTE

Some options are only available in UWP apps.

Debugger to Launch

Specifies the default host for the debugger.

- Select **Local Machine** to start the application on the Visual Studio host computer. For more information, see [Running apps on the local machine](#).
- Select **Simulator** to start the application in the Simulator. For more information, see [Running apps in the simulator](#).
- Select **Remote Machine** to start the application on a remote computer. For more information about remote debugging, see [Running apps on a remote machine](#).

Launch Application

Specifies whether to start the application when you press **F5** or click **Debug > Start Debugging**. Select **Yes** to start the application; otherwise, select **No**. If you select **No**, you can still debug the application if you use a different method to start it.

Debugger Type

Specifies the types of code to debug. Select **Script Only** to debug JavaScript code. Select **Managed Only** to debug code that is managed by the common language runtime. Select **Native Only** to debug C++ code. Select **Native with Script** to debug C++ and JavaScript. Select **Mixed (Managed and Native)** to debug both managed and C++ code.

Allow Local Network Loopback

Specifies whether access to the IP loopback address is allowed for app testing. Select **Yes** to allow use of the loopback address if the client app is on the same machine where the server application is running; otherwise, select **No**. This property is available only if the **Debugger to Launch** property is set to **Remote Machine**.

Machine Name

Specifies the name of the remote computer to host the debugger. This property is available only if **Debugger to Launch** is set to **Remote Machine**.

Require Authentication

Specifies whether the remote computer requires authentication. This property is available only if **Debugger to Launch** is set to **Remote Machine**.

Properties window

10/18/2019 • 2 minutes to read • [Edit Online](#)

Use this window to view and change the design-time properties and events of selected objects that are located in editors and designers. You can also use the **Properties** window to edit and view file, project, and solution properties. You can find **Properties** Window on the **View** menu. You can also open it by pressing **F4** or by typing **Properties** in the search box.

The **Properties** window displays different types of editing fields, depending on the needs of a particular property. These edit fields include edit boxes, drop-down lists, and links to custom editor dialog boxes. Properties shown in gray are read-only.

UIElement List

Object name

Lists the currently selected object or objects. Only objects from the active editor or designer are visible. When you select multiple objects, only properties common to all selected objects appear.

Categorized

Lists all properties and property values for the selected object, by category. You can collapse a category to reduce the number of visible properties. When you expand or collapse a category, you see a plus (+) or minus (-) to the left of the category name. Categories are listed alphabetically.

Alphabetical

Alphabetically sorts all design-time properties and events for selected objects. To edit an undimmed property, click in the cell to its right and enter changes.

Property Pages

Displays the **Property Pages** dialog box or **Project Designer** for the selected item. Property Pages displays a subset, the same or a superset of the properties available in the **Properties** window. Use this button to view and edit properties related to your project's active configuration.

Properties

Displays the properties for an object. Many objects also have events that can be viewed using the **Properties** window.

Sort by Property Source

Groups properties by source, such as inheritance, applied styles, and bindings. Only available when editing XAML files in the designer.

Events

Displays the events for an object.

NOTE

This **Properties** window toolbar control is only available when a form or control designer is active in the context of a Visual C# project. When editing XAML files, events appear on a separate tab of the properties window.

Messages

Lists all Windows messages. Allows you to add or delete specified handler functions for the messages provided for the selected class.

NOTE

This **Properties** window toolbar control is only available when **Class View** is the active window in the context of a Visual C++ project.

Overrides

Lists all virtual functions for the selected class and allows you to add or delete overriding functions.

NOTE

This **Properties** window toolbar control is only available when **Class View** is the active window in the context of a Visual C++ project.

Description pane

Shows the property type and a short description of the property. You can turn the description of the property off and on using the Description command on the shortcut menu.

NOTE

This **Properties** window toolbar control is not available when editing XAML files in the designer.

Thumbnail view

Shows a visual representation of the currently selected element when editing XAML files in the designer.

Search

Provides a Search function for properties and events when editing XAML files in the designer. The search box responds to partial word searches and updates search results as you type.

See also

- [Project Properties Reference](#)
- [Customizing window layouts](#)

Team Explorer reference

11/26/2019 • 2 minutes to read • [Edit Online](#)

This article provides links to Azure DevOps articles about the various functions in **Team Explorer**.

Use the **Team Explorer** tool window to coordinate your code efforts with other team members to develop a project, and to manage work that's assigned to you, your team, or your projects. **Team Explorer** connects Visual Studio to Git and GitHub repositories, Team Foundation version control (TFVC) repositories, and projects hosted on [Azure DevOps Services](#) or an on-premises [Azure DevOps Server](#) (formerly known as TFS). You can manage source code, work items, and builds.

Home page

After you [connect to a project](#) in **Team Explorer**, the following links become available in the **Project** section:

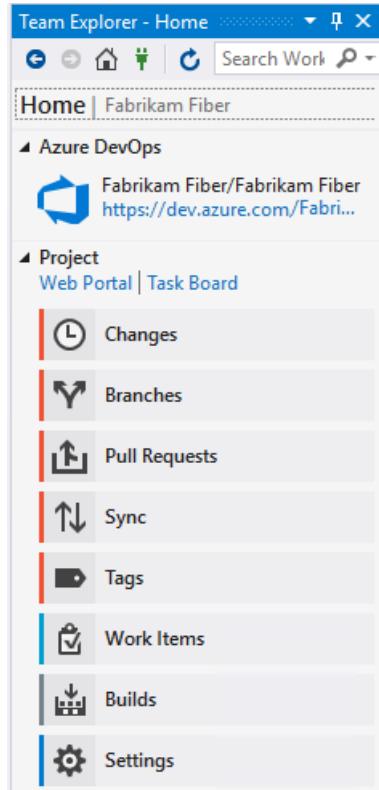
- [Clone repository](#)
- [Web Portal](#)
- [Task Board](#)

The **Home** page has different functions depending on whether you're connected to a [Git](#) or [Team Foundation Version Control \(TFVC\)](#) repository.

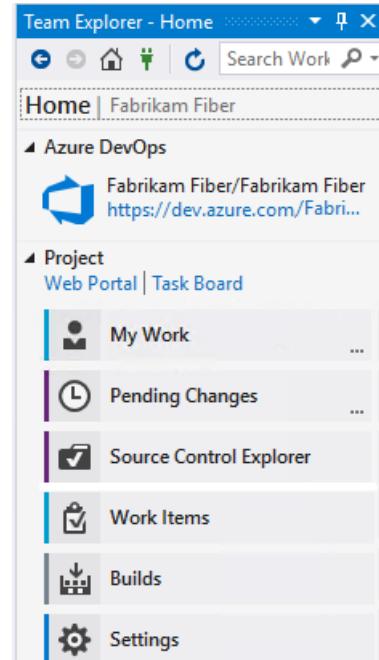
TIP

For a comparison of the two version control systems, see [Choose the right version control for your project \(Azure DevOps\)](#).

HOME PAGE WITH GIT



HOME PAGE WITH TFVC



Changes page (Git)

See [Save work with commits](#).

Branches page (Git)

See [Create work in branches](#).

Pull Requests page (Git)

See [Review code with pull requests](#).

Sync page (Git)

See [Update code with fetch and pull](#).

Tags page (Git)

See [Work with Git tags](#).

My Work page (TFVC)

See [Suspend/resume work](#) and [Code review](#).

Pending Changes page (TFVC)

See [Manage pending changes](#), [Find shelvesets](#), and [Resolve conflicts](#).

Source Control Explorer page (TFVC)

See [Add/view files and folders](#).

Work Items page

The **Work Items** page lets you see [work item](#) queries. See:

- [Add work items](#)
- [Use the query editor to list and manage queries](#)
- [Organize query folders and set query permissions](#)
- [Open query in Excel](#)
- [Open query in Project](#)
- [Email query results list using Outlook](#)
- [Create reports from query in Excel \(TFS only\)](#)

NOTE

There's a new [Work Items experience](#) in Visual Studio 2019. For information about viewing work items in Visual Studio 2019, see [View and add work items](#).

Builds page

The **Builds** page lets you see build definitions for the project.

See:

- [Create build pipelines](#)
- [View and manage builds](#)
- [Manage the build queue](#)
- [Install continuous delivery \(CD\) tools for Visual Studio](#)
- [Configure and execute continuous delivery \(CD\) for your app](#)

Settings page

The **Settings** page lets you configure administrative features for either a project or project collection. See the following articles:

PROJECT	PROJECT COLLECTION	OTHER
Security, Group Membership Security, Source Control (TFVC) Work Item Areas Work Item Iterations Portal Settings Project Alerts	Security, Group Membership Source Control (TFVC) Process Template Manager	Git Global Settings Git Repository Settings

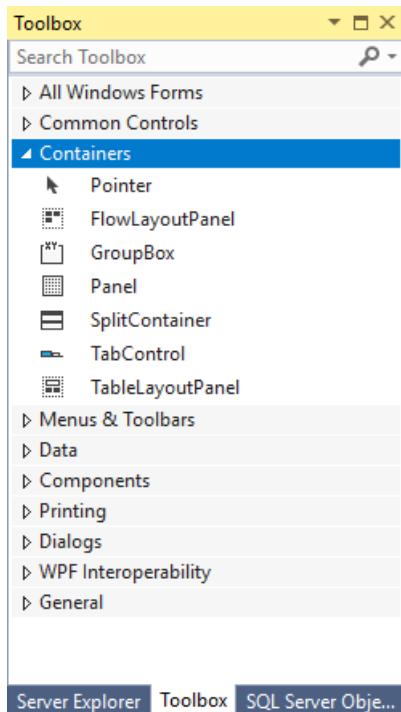
See also

- [Connect to projects in Team Explorer](#)

Toolbox

10/18/2019 • 2 minutes to read • [Edit Online](#)

The **Toolbox** window displays controls that you can add to Visual Studio projects. To open Toolbox, choose **Toolbox** on the **View** menu.



You can drag and drop different controls onto the surface of the designer you are using, and resize and position the controls.

Toolbox appears in conjunction with designer views, such as the designer view of a XAML file. **Toolbox** displays only those controls that can be used in the current designer. You can search within **Toolbox** to further filter the items that appear.

NOTE

For some project types, **Toolbox** may not show any items.

The .NET version that your project targets also affects the set of controls visible in Toolbox. You can change the target framework version from the project's property pages, if necessary. Select the project node in **Solution Explorer**, and then on the menu bar, choose **Project > projectname Properties**. On the **Application** tab, use the **Target framework** drop-down.

Manage the Toolbox window and its controls

By default **Toolbox** is collapsed along the left side of the Visual Studio IDE, and appears when the cursor is moved over it. You can pin **Toolbox** (by clicking the **Pin** icon on its toolbar) so that it remains open when you move the cursor. You can also undock the **Toolbox** window and drag it anywhere on your screen. You can dock, undock, and hide **Toolbox** by right-clicking its toolbar and selecting one of the options.

You can rearrange the items in a **Toolbox** tab or add custom tabs and items by using the following commands on the right-click menu:

- **Rename item** - Renames the selected item.
- **Show All** - Shows all possible controls (not just the ones that apply to the current designer).
- **List View** - Shows the controls in a vertical list. If unchecked, the controls appear horizontally.
- **Choose Items** - Opens the **Choose Toolbox Items** dialog box so that you can specify the items that appear in the **Toolbox**. You can show or hide an item by selecting or clearing its check box.
- **Sort items alphabetically** - Sorts the items by name.
- **Reset Toolbar** - Restores the default **Toolbox** settings and items.
- **Add Tab** - Adds a new **Toolbox** tab.
- **Move Up** - Moves the selected item up.
- **Move Down** - Moves the selected item down.

Create and distribute custom Toolbox controls

You can create custom **Toolbox** controls, starting either with a project template that's based on [Windows Presentation Foundation](#) or on [Windows Forms](#). You can then distribute your custom control to your teammates, or publish it on the web by using the [Toolbox Controls Installer](#).

Help on Toolbox tabs

The following topics provide more information about some of the available **Toolbox** tabs:

- [Toolbox, Data Tab](#)
- [Toolbox, Components Tab](#)
- [Toolbox, HTML Tab](#)

See also

- [Choose Toolbox items, WPF components](#)

Toolbox, Components tab

10/18/2019 • 2 minutes to read • [Edit Online](#)

Displays components you can add to Visual Basic and C# designers for Windows Forms. In addition to the .NET components that are included with Visual Studio, such as the [MessageQueue](#) and [EventLog](#) components, you can add your own or third-party components to this tab.

To display this tab, open a Windows Forms designer. Select **View > Toolbox**. In **Toolbox**, select the **Components** tab.

Components

BackgroundWorker

Creates a [BackgroundWorker](#) component instance that can run an operation on a separate, dedicated thread. For more information, see [BackgroundWorker component](#).

DirectoryEntry

Creates a [DirectoryEntry](#) component instance that encapsulates a node or object in the Active Directory hierarchy and can be used to interact with Active Directory service providers.

DirectorySearcher

Creates a [DirectorySearcher](#) component instance that you can use to perform queries against the Active Directory.

ErrorProvider

Creates a [ErrorProvider](#) component instance, which indicates to the end user that a control on a form has an error associated with it. For more information, see [ErrorProvider component](#).

EventLog

Creates an [EventLog](#) component instance you can use to interact with system and custom event logs, including writing events to a log and reading log data.

FileSystemWatcher

Creates a [FileSystemWatcher](#) component instance that you can use to monitor for changes to any directory or file to which you have access.

HelpProvider

Creates a [HelpProvider](#) component instance that provides pop-up or online help for controls. For more information, see [HelpProvider component](#).

ImageList

Creates a [ImageList](#) component instance that provides methods to manage a collection of [Image](#) objects. For more information, see [ImageList component](#).

MessageQueue

Creates a [MessageQueue](#) component instance that you can use to interact with message queues, including reading messages from and writing messages to queues, processing transactions, and performing queue administration tasks.

PerformanceCounter

Creates a [PerformanceCounter](#) component instance that you can use to interact with Windows performance counters, including creating new categories and instances, reading values from counters, and performing calculations on counter data.

Process

Creates a [Process](#) component instance you can use to stop, start, and manipulate the data associated with processes on your system.

SerialPort

Creates a [SerialPort](#) component instance that provides synchronous and event-driven I/O, access to pin and break states, and access to serial driver properties.

ServiceController

Creates a [ServiceController](#) component instance you can use to manipulate existing services, including starting and stopping services and sending commands to them.

Timer

Creates a [Timer](#) component instance you can use to add time-based functionality to your Windows-based applications. For more information, see [Timer component](#).

NOTE

There is also a system-based [Timer](#) that you can add to the [Toolbox](#). This [Timer](#) is optimized for server applications, and the Windows Forms [Timer](#) is best suited for use on Windows Forms.

See also

- [Controls to use on Windows Forms](#)
- [Choose Toolbox items, WPF components](#)
- [Toolbox](#)

Toolbox, Data tab

10/18/2019 • 2 minutes to read • [Edit Online](#)

Displays data objects you can add to a forms and components. The **Data** tab of the **Toolbox** appears when you create a project that has an associated designer. The **Toolbox** appears by default in the Visual Studio integrated development environment; if you need to display the **Toolbox**, select **Toolbox** from the **View** menu.

TIP

Running the Data Source Configuration Wizard automatically creates and configures most data items. For more information, see [Add new data sources](#).

UI Element list

To go directly to the .NET reference page for a component, press **F1** on the item in the **Toolbox** or on the component item in the tray of the designer.

NAME	DESCRIPTION
DataSet	Adds an instance of a typed or untyped dataset to the form or component. When you drag this object onto a designer, it displays a dialog box that allows you to select an existing typed dataset class or specify that you want to create a new, blank, untyped dataset. Note: You do not use the DataSet object on the Toolbox to create a new typed dataset schema and class. For more information, see Create and configure datasets .
DataGridView	Provides a powerful and flexible way to display data in a tabular format.
BindingSource	Simplifies the process of binding controls to an underlying data source.
BindingNavigator	Represents the navigation and manipulation user interface (UI) for controls on a form that are bound to data.

See also

- [Access Data in Visual Studio](#)
- [Visual Studio data tools for .NET](#)
- [Dataset tools in Visual Studio](#)
- [Bind controls to data in Visual Studio](#)
- [Bind Windows Forms controls to data in Visual Studio](#)
- [Edit data in datasets](#)
- [Validate data in datasets](#)

Toolbox, HTML tab

10/18/2019 • 5 minutes to read • [Edit Online](#)

The **HTML** tab of the Toolbox provides components that are useful on web pages and web forms. To view this tab, first open a document for editing in the HTML designer. On the **View** menu, click **Toolbox**, and then click the **HTML** tab of the Toolbox.

To create an instance of a tool on the **HTML** tab, either double-click the tool to add it to your document at the current insertion point, or select the tool and drag it to the desired position on the editing surface.

UI elements

The following tools are available by default on the HTML tab.

Pointer



This tool is selected by default when any Toolbox tab opens. It cannot be deleted. The pointer enables you to drag objects onto the Design view surface, resize them, and reposition them on the page or form. For more information, see [Toolbox](#).

Input (Button)



Inserts an `input` element of `type="button"`. To change the text that is displayed, edit the `name` property. By default, `id="Button1"` is inserted for the first button, `id="Button2"` for the second, and so on.

When you drag **Input (Button)** onto the Design view surface, HTML markup like the following is inserted into your document:

```
<input id="Button1" type="button" value="Button" name="Button1">
```

Input (Reset)



Inserts an `input` element of `type="reset"`. To change the text that is displayed, edit the `name` property. By default, `id="Reset1"` is inserted for the first reset button, `id="Reset2"` for the second, and so on.

When you drag **Input (Reset)** onto the Design view surface, HTML markup like the following is inserted into your document:

```
<input id="Reset1" type="reset" value="Reset" name="Reset1">
```

Input (Submit)



Inserts an `input` element of `type="submit"`. To change the text that is displayed, edit the `name` property. By default, `id="Submit1"` is inserted for the first submit button, `id="Submit2"` for the second, and so on.

When you drag **Input (Submit)** onto the Design view surface, HTML markup like the following is inserted into

your document:

```
<input id="Submit1" type="submit" value="Submit" name="Submit1">
```

Input (Text)

 **Text Field**

Inserts an `input` element of `type="text"` in your document. To change the default text that is displayed, edit the `value` attribute. By default, `id="Text1"` is inserted for the first text field, `id="Text2"` for the second, and so on.

When you drag **Input (Text)** onto the Design view surface, HTML markup like the following is inserted into your document:

```
<input id="Text1" TYPE="text" value="Text Field" name="Text1">
```

IMPORTANT

It is recommended that you validate all user input. For more information, see [Validating User Input in ASP.NET Web Pages \(Razor\) Sites](#).

Input (File)

 **File Field**

Inserts an `input` element of `type="file"` in your document. By default, `id="File1"` is inserted for the first file field, `id="File2"` for the second, and so on.

When you drag **Input (File)** onto the Design view surface, HTML markup like the following is inserted into your document:

```
<input id="File1" type="file" name="File1">
```

IMPORTANT

It is recommended that you validate all user input. For more information, see [Validating User Input in ASP.NET Web Pages \(Razor\) Sites](#).

Input (Password)

 **Password Field**

Inserts an `input` element of `type="password"`. By default, `id="Password1"` is inserted for the first password field, `id="Password2"` for the second, and so on.

When you drag **Input (Password)** onto the Design view surface, HTML markup like the following is inserted into your document:

```
<input id="Password1" type="password" name="Password1">
```

IMPORTANT

If your application transmits user names and passwords, you should configure your website to use Secure Sockets Layer (SSL) to encrypt the transmission. For more information, see [Securing Connections](#). Additionally, it is recommended that you validate all user input. For more information, see [Validating User Input in ASP.NET Web Pages \(Razor\) Sites](#).

Input (Check box)



Inserts an `input` element of `type="checkbox"`. To change the text that is displayed, edit the `name` property. By default, `id="Checkbox1"` is inserted for the first check box, `id="Checkbox2"` for the second, and so on.

When you drag **Input (Check box)** onto the Design view surface, HTML markup like the following is inserted into your document:

```
<input id="Checkbox1" type="checkbox" name="Checkbox1">
```

Input (Radio)



Inserts an `input` element of `type="radio"`. To change the text that is displayed, edit the `name` property. By default, `id="Radio1"` is inserted for the first radio button, `id="Radio2"` for the second, and so on.

When you drag **Input (Radio)** onto the Design view surface, HTML markup like the following is inserted into your document:

```
<input id="Radio1" type="radio" name="Radio1">
```

Input (Hidden)



Inserts an `input` element of `type="hidden"`. By default, `id="Hidden1"` is inserted for the first hidden field, `id="Hidden2"` for the second, and so on.

When you drag **Input (Hidden)** onto the Design view surface, HTML markup like the following is inserted into your document:

```
<input id="Hidden1" type="hidden" name="Hidden1">
```

Textarea



Inserts a `textarea` element. You can resize the text area, or use its scroll bars to view text that extends beyond its display area. To change the default text that is displayed, edit the `value` attribute. By default, `id="textarea1"` is inserted for the first text area, `id="textarea 2"` for the second, and so on.

When you drag **Textarea** onto the Design view surface, HTML markup like the following is inserted into your document:

```
<textarea id="textarea 1" name="textarea 1" rows=2 cols=20></textarea>
```

IMPORTANT

It is recommended that you validate all user input. For more information, see [Validating User Input in ASP.NET Web Pages \(Razor\) Sites](#).

Table



Inserts a `table` element.

When you drag **Table** onto the Design view surface, HTML markup like the following is inserted into your document:

```
<table cellspacing="1" width="75%" border=1> <tr><td></td></tr></table>
```

Image



Inserts an `img` element. Edit this element to specify its `src` and its `alt` text.

When you drag **Image** onto the Design view surface, HTML markup like the following is inserted into your document:

```
<img alt="" src="">
```

Select



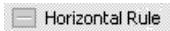
Inserts a dropdown `select` element (without a `size` attribute). By default, `id="select1"` is inserted for the first list box, `id="select2"` for the second, and so on.

When you drag **Select** onto the Design view surface, HTML markup like the following is inserted into your document:

```
<select id="select1" name="select1"><option selected></option></select>
```

You can create a multi-line `select` element by increasing the value of the `size` property.

Horizontal Rule



Inserts an `hr` element. To increase the thickness of the line, edit the `size` attribute.

When you drag **Horizontal Rule** onto the Design view surface, HTML markup like the following is inserted into your document:

```
<hr width="100%" size=1>
```

Div



Inserts a `div` element that includes an `ms_positioning="FlowLayout"` attribute. Except for the width and height, this item is identical to a Flow Layout Panel. To format the text that is contained within the `div` element, add a `class="stylename"` attribute to the opening tag.

When you drag **Div** onto the Design view surface, HTML markup like the following is inserted into your document:

```
<div ms_positioning="FlowLayout" style="width: 70px; position: relative; height: 15px">Label</div>
```

See also

- [Toolbox](#)

Devenv command-line switches

10/18/2019 • 4 minutes to read • [Edit Online](#)

Devenv lets you set various options for the IDE, build projects, debug projects, and deploy projects from the command line. Use these switches to run the IDE from a script or a .bat file (such as a nightly build script), or to start the IDE in a particular configuration.

NOTE

For build-related tasks, it's recommended that you use MSBuild instead of devenv. For more information, see [MSBuild command-line reference](#).

For information about switches that are related to VSPackage development, also see [Devenv command-line switches for VSPackage development](#).

Devenv switch syntax

Commands that begin with `devenv` are handled by the `devenv.com` utility, which delivers output through standard system streams, such as `stdout` and `stderr`. The utility determines the appropriate I/O redirection when it captures output, for example to a .txt file.

Alternatively, commands that begin with `devenv.exe` can use the same switches, but the `devenv.com` utility is bypassed. Using `devenv.exe` directly prevents output from appearing on the console.

The syntax rules for `devenv` switches resemble the rules for other DOS command-line utilities. The following syntax rules apply to all `devenv` switches and their arguments:

- Commands begin with `devenv`.
- Switches aren't case-sensitive.
- You can specify a switch by using a hyphen ("–") or a forward slash ("/").
- When specifying a solution or project, the first argument is the name of the solution file or project file, including file path.
- If the first argument is a file that's not a solution or project, that file opens in the appropriate editor, in a new instance of the IDE.
- When you supply a project file name instead of a solution file name, a `devenv` command searches the parent folder of the project file for a solution file that has the same name. For example, the command `devenv myproject1.vbproj /build` searches the parent folder for a solution file that's named `myproject1.sln`.

NOTE

One and only one solution file that references this project should be located in its parent folder. If the parent folder contains no solution file that references this project, or if the parent folder contains two or more solution files that reference it, then a temporary solution file is created.

- When file paths and file names include spaces, you must enclose them in quotation marks (""). For example, `"c:\project a\"`.

- Insert one space character between switches and arguments on the same line. For example, the command `devenv /log output.txt` opens the IDE and outputs all log information for that session to `output.txt`.
- You can't use pattern-matching syntax in `devenv` commands.

Devenv switches

The following command-line switches display the IDE and do the described task.

COMMAND LINE SWITCH	DESCRIPTION
/Command	Starts the IDE and executes the specified command. <code>devenv /command "nav https://docs.microsoft.com/"</code>
/DebugExe	Loads a C++ executable under the control of the debugger. This switch isn't available for Visual Basic or C# executables. For more information, see Automatically start a process in the debugger . <code>devenv /debugexe mysln.exe</code>
/Diff	Compares two files. Takes four parameters: <i>SourceFile</i> , <i>TargetFile</i> , <i>SourceDisplayName</i> (optional), and <i>TargetDisplayName</i> (optional). <code>devenv /diff File1 File2 Alias1 Alias2</code>
/DoNotLoadProjects	Opens the specified solution without loading any projects. <code>devenv /donotloadprojects mysln.sln</code>
/Edit	Opens the specified files in a running instance of this application. If there are no running instances, it starts a new instance with a simplified window layout. <code>devenv /edit File1 File2</code>
/LCID or /L	Sets the default language for the IDE. If the specified language isn't included in your installation of Visual Studio, this setting is ignored. <code>devenv /l 1033</code>
/Log	Starts Visual Studio and logs all activity to the log file. <code>devenv /log mylogfile.xml</code>
/NoSplash	Opens the IDE without showing the splash screen. <code>devenv /nosplash File1 File2</code>
/Run or /R	Compiles and runs the specified solution. <code>devenv /run mysln.sln</code>

COMMAND LINE SWITCH	DESCRIPTION
/RunExit	Compiles and runs the specified solution, minimizes the IDE when the solution is run, and closes the IDE after the solution has finished running. <code>devenv /runexit mysln.sln</code>
/SafeMode	Starts Visual Studio in safe mode. This switch loads only the default environment, the default services, and the shipped versions of third-party packages. This switch takes no arguments.
/UseEnv	Causes the IDE to use PATH, INCLUDE, LIBPATH, and LIB environment variables for C++ compilation. This switch is installed with the Desktop development with C++ workload. For more information, see Setting the Path and Environment Variables for Command-Line Builds .

The following command-line switches don't display the IDE.

COMMAND LINE SWITCH	DESCRIPTION
/?	Displays help for <code>devenv</code> switches in the Command Prompt window . This switch takes no arguments.
/Build	Builds the specified solution or project according to the configuration of the specified solution. <code>devenv mysln.sln /build</code>
/Clean	Deletes any files created by the build command, without affecting source files. <code>devenv mysln.sln /clean</code>
/Deploy	Builds the solution, along with files necessary for deployment, according to the solution's configuration. <code>devenv mysln.sln /deploy</code>
/Out	Lets you specify a file to receive errors when you build. <code>devenv mysln.sln /build Debug /out log.txt</code>
/Project	The project to build, clean, or deploy. You can use this switch only if you've also supplied the <code>/Build</code> , <code>/Rebuild</code> , <code>/Clean</code> , or <code>/Deploy</code> switch. <code>devenv mysln.sln /build Debug /project proj1</code>

COMMAND LINE SWITCH	DESCRIPTION
/ProjectConfig	<p>Specifies the project configuration to build or deploy. You can use this switch only if you've also supplied the /Project switch.</p> <pre data-bbox="837 309 1425 361">devenv mysln.sln /build Release /project proj1 /projectconfig Release</pre>
/Rebuild	<p>Cleans and then builds the specified solution or project according to the configuration of the specified solution.</p> <pre data-bbox="837 503 1139 532">devenv mysln.sln /rebuild</pre>
/ResetSettings	<p>Restores Visual Studio default settings. Optionally resets the settings to the specified .vssettings file.</p> <pre data-bbox="837 676 1334 705">devenv /resetsettings mysettings.vssettings</pre>
/Upgrade	<p>Upgrades the specified solution file and all its project files, or the specified project file, to the current Visual Studio formats for these files.</p> <pre data-bbox="837 893 1139 923">devenv mysln.sln /upgrade</pre>

See also

- [General, Environment, Options Dialog Box](#)
- [Devenv command-line switches for VSPackage development](#)

/? (devenv.exe)

10/18/2019 • 2 minutes to read • [Edit Online](#)

Displays a message box listing all `devenv` switches, with a brief description of each switch.

Syntax

```
devenv /?
```

See also

- [Devenv command-line switches](#)

/Build (devenv.exe)

10/18/2019 • 2 minutes to read • [Edit Online](#)

Builds a solution or project using a specified solution configuration file.

Syntax

```
devenv SolutionName /Build [SolnConfigName [/Project ProjName [/ProjectConfig ProjConfigName]] [/Out OutputFilename]]
```

Arguments

- *SolutionName*

Required. The full path and name of the solution file.

- *SolnConfigName*

Optional. The name of the solution configuration (such as `Debug` or `Release`) to be used to build the solution named in *SolutionName*. If multiple solution platforms are available, you must also specify the platform (for example, `Debug|Win32`). If this argument is unspecified or an empty string (`" "`), the tool uses the solution's active configuration.

- `/Project` *ProjName*

Optional. The path and name of a project file within the solution. You can enter a relative path from the *SolutionName* folder to the project file, or the project's display name, or the full path and name of the project file.

- `/ProjectConfig` *ProjConfigName*

Optional. The name of a project build configuration (such as `Debug` or `Release`) to be used when building the named project. If more than one solution platform is available, you must also specify the platform (for example, `Debug|Win32`). If this switch is specified, it overrides the *SolnConfigName* argument.

- `/out` *OutputFilename*

Optional. The name of a file that you want to send the tool's output to. If the file already exists, the tool appends the output to the end of the file.

Remarks

- The `/Build` switch performs the same function as the **Build Solution** menu command within the integrated development environment (IDE).
- Enclose strings that include spaces in double quotes.
- Summary information for builds, including errors, can be displayed in the command window, or in any log file specified with the `/out` switch.
- The `/Build` switch only builds projects that have changed since the last build. To build all projects in a solution, use [/rebuild](#) instead.

- If you get an error message that says **Invalid project configuration**, make sure that you've specified a solution platform or project platform (for example, `Debug|Win32`).

Example

The following command builds the project `csharpWinApp`, using the `Debug` project build configuration within `MySolution`.

```
devenv "%USERPROFILE%\source\repos\MySolution.sln" /build Debug /project "CSharpWinApp\CSharpWinApp.csproj"  
/projectconfig Debug
```

See also

- [Build and clean projects and solutions](#)
- [Devenv command-line switches](#)
- [/Rebuild \(devenv.exe\)](#)
- [/Clean \(devenv.exe\)](#)
- [/Out \(devenv.exe\)](#)

/Clean (devenv.exe)

10/18/2019 • 2 minutes to read • [Edit Online](#)

Cleans all intermediary files and output directories.

Syntax

```
devenv SolutionName /Clean [Config [/Project ProjName [/ProjectConfig ProjConfigName]] [/Out OutputFilename]]
```

Arguments

- *SolutionName*

Required. The full path and name of the solution file.

- *Config*

Optional. The configuration (such as `Debug` or `Release`) to clean the intermediary files for the solution named in *SolutionName*. If more than one solution platform is available, you must also specify the platform (for example, `Debug|Win32`). If this argument is unspecified or an empty string (`""`), the tool uses the solution's active configuration.

- `/Project` *ProjName*

Optional. The path and name of a project file within the solution. You can enter the project's display name or a relative path from the *SolutionName* folder to the project file. You can also enter the full path and name of the project file.

- `/ProjectConfig` *ProjConfigName*

Optional. The project's build configuration name (such as `Debug` or `Release`) to be used when cleaning the `/Project` named. If more than one solution platform is available, you must also specify the platform (for example, `Debug|Win32`). If this switch is specified, it overrides the *Config* argument.

- `/out` *OutputFilename*

Optional. The name of a file that you want to send the tool's output to. If the file already exists, the tool appends the output to the end of the file.

Remarks

This switch does the same function as the **Clean Solution** menu command within the IDE.

Enclose strings that include spaces in double quotation marks.

Summary information when cleaning and building, including errors, can be displayed in the **Command** window, or in any log file specified with the `/Out` switch.

If the `/Project` switch isn't specified, the cleaning action is done on all projects in the solution, even if *FileName* was specified as a project file.

Example

The first example cleans the `MySolution` solution, using the default configuration specified in the solution file.

The second example cleans the project `CSharpWinApp`, using the `Debug` project build configuration within `MySolution`.

```
devenv "%USERPROFILE%\source\repos\MySolution\MySolution.sln" /Clean  
devenv "%USERPROFILE%\source\repos\MySolution\MySolution.sln" /Clean "Debug" /project  
"CSharpWinApp\CSharpWinApp.csproj" /projectconfig "Debug"
```

See also

- [Devenv command-line switches](#)
- [/Build \(devenv.exe\)](#)
- [/Rebuild \(devenv.exe\)](#)
- [/Out \(devenv.exe\)](#)

/Command (devenv.exe)

10/18/2019 • 2 minutes to read • [Edit Online](#)

Executes the specified command after launching the Visual Studio IDE.

Syntax

```
devenv /Command CommandName
```

Arguments

CommandName

Required. The complete name of a Visual Studio command or its alias, enclosed in double quotation marks. For more information about command and alias syntax, see [Visual Studio Commands](#).

Remarks

After startup is complete, the IDE executes the named command.

If you use this switch, the IDE doesn't display the Start Page on startup.

If an add-in exposes a command, you can use this switch to launch the add-in from the command line. For more information, see [How to: Control add-ins by using the add-in manager](#).

Example

The first example launches Visual Studio and automatically runs the macro Open Favorite Files.

The second example opens a web browsing tab within the IDE and navigates to the Microsoft Docs site.

The third example creates a new file called `some_file.cs` and opens it in a code editor.

```
devenv /command "Macros.MyMacros.Module1.OpenFavoriteFiles"  
devenv /command "navigate https://docs.microsoft.com/"  
devenv /command "nf some_file.cs"
```

See also

- [Devenv command-line switches](#)
- [Visual Studio Command Aliases](#)
- [Command window](#)

/DebugExe (devenv.exe)

10/18/2019 • 2 minutes to read • [Edit Online](#)

Opens the specified executable file to be debugged.

Syntax

```
devenv /DebugExe ExecutableFile
```

Arguments

- *ExecutableFile*

Required. The path and file name of an `.exe` file. If the `.exe` file isn't found or doesn't exist, no warning or error is displayed, and Visual Studio starts normally.

Remarks

Any strings following the *ExecutableFile* parameter are passed to that file as arguments.

Example

The following example opens the file `MyApplication.exe` for debugging.

```
devenv /debugexe MyApplication.exe
```

See also

- [Devenv command-line switches](#)

/Deploy (devenv.exe)

10/18/2019 • 2 minutes to read • [Edit Online](#)

Deploys a solution after a build or rebuild. Applies to managed code projects only.

Syntax

```
devenv SolutionName /Deploy [SolnConfigName [/Project ProjName [/ProjectConfig ProjConfigName]] [/Out OutputFilename]]
```

Arguments

- *SolutionName*

Required. The full path and name of the solution file.

- *SolnConfigName*

Optional. The name of the solution configuration (such as `Debug` or `Release`) to be used to build the solution named in *SolutionName*. If more than one solution platform is available, you must also specify the platform (for example, `Debug|Win32`). If this argument is unspecified or an empty string (`""`), the tool uses the solution's active configuration.

- `/Project` *ProjName*

Optional. The path and name of a project file within the solution. You can enter the project's display name or a relative path from the *SolutionName* folder to the project file. You can also enter the full path and name of the project file.

- `/ProjectConfig` *ProjConfigName*

Optional. The names of a project build configuration (such as `Debug` or `Release`) to be used when building the `/Project` named. If more than one solution platform is available, you must also specify the platform (for example, `Debug|Win32`). If this switch is specified, it overrides the *SolnConfigName* argument.

- `/out` *OutputFilename*

Optional. The name of a file that you want to send the tool's output to. If the file already exists, the tool appends the output to the end of the file.

Remarks

The specified project must be a deployment project. If the specified project isn't a deployment project, when the project that has been built is passed for deployment, it fails with an error.

Enclose strings that include spaces in double quotation marks.

Summary information for builds, including errors, can be displayed in the **Command** window, or in any log file specified with the `/Out` switch.

Example

This example deploys the project `csharpWinApp`, using the `Release` project build configuration within `MySolution`.

```
devenv "%USERPROFILE%\source\repos\MySolution\MySolution.sln" /deploy Release /project  
"CSharpWinApp\CSharpWinApp.csproj" /projectconfig Release
```

See also

- [Devenv command-line switches](#)
- [/Project \(devenv.exe\)](#)
- [/Build \(devenv.exe\)](#)
- [/Clean \(devenv.exe\)](#)
- [/Rebuild \(devenv.exe\)](#)
- [/Out \(devenv.exe\)](#)

/Diff (devenv.exe)

10/18/2019 • 2 minutes to read • [Edit Online](#)

Compares two files. The differences are displayed in a special Visual Studio window.

Syntax

```
devenv /Diff SourceFile TargetFile [SourceDisplayName [TargetDisplayName]]
```

Arguments

- *SourceFile*

Required. The full path and name of the first file to be compared.

- *TargetFile*

Required. The full path and name of the second file to be compared.

- *SourceDisplayName*

Optional. The display name of the first file.

- *TargetDisplayName*

Optional. The display name of the second file.

Remarks

If an instance of the IDE is already open, the file comparison appears in a tab in the current IDE.

Example

The first example compares two files without changing their display names. The second example compares the files while changing both of their display names. The third and fourth examples compare two files but apply an alias to only the first file or the second file.

```
devenv /diff File1.txt File2.txt  
devenv /diff File1.txt File2.txt FirstAlias "Second Alias"  
devenv /diff File1.txt File2.txt "File One"  
devenv /diff File1.txt File2.txt "" FileTwo
```

See also

- [Devenv command-line switches](#)

/DoNotLoadProjects (devenv.exe)

10/18/2019 • 2 minutes to read • [Edit Online](#)

New for Visual Studio 2019 version 16.1

Opens the specified solution without loading any projects. For more information, see [Filtered solutions in Visual Studio](#).

Syntax

```
devenv /DoNotLoadProjects SolutionName
```

Arguments

SolutionName

Required. The full path and name of the solution to be opened.

Example

The example opens the solution MySln.sln without loading any projects.

```
devenv /donotloadprojects MySln.sln
```

See also

- [Filtered solutions in Visual Studio](#)
- [Devenv command-line switches](#)

/Edit (devenv.exe)

10/18/2019 • 2 minutes to read • [Edit Online](#)

Opens the specified file in an existing instance of Visual Studio.

Syntax

```
devenv /Edit [File1[ FileN]...]
```

Arguments

- *File1*

Optional. The file to open in an existing instance of Visual Studio. If no instance of Visual Studio exists, a new instance is created with a simplified window layout, and the tool opens *File1* in the new instance.

- *FileN*

Optional. One or more additional files to open in the existing instance of Visual Studio.

Remarks

When a file isn't specified, an existing Visual Studio instance receives focus. If no file is specified and no instance of Visual Studio exists, the tool creates an instance with a simplified window layout.

If the existing Visual Studio instance is in a modal state, the file opens in the existing instance when Visual Studio exits the modal state. For example, this situation may occur when the [Options dialog box](#) is open.

If more than one instance of Visual Studio is open, the file is opened in the most recently opened instance.

Example

The first example opens the file `MyFile.cs` in an existing instance of Visual Studio. If a Visual Studio instance doesn't exist, the tool opens the file in a new instance. The second example is similar except that it opens three files instead of just one file.

```
devenv /edit MyFile.cs  
devenv /edit MyFile1.cs MyFile2.cs MyFile3.cs
```

See also

- [Devenv command-line switches](#)

/LCID (devenv.exe)

10/18/2019 • 2 minutes to read • [Edit Online](#)

Sets the default language used for text, currency, and other values within the IDE.

Syntax

```
devenv {/LCID|/L} LocaleID
```

Arguments

- *LocaleID*

Required. The locale identifier (LCID) of the language you specify.

Remarks

Loads the IDE and sets the default natural language for the environment. This change is persisted between sessions, and the IDE shows this change in the **Tools > Options > Environment > International Settings > Language** box.

If the specified language isn't available on your system, the `/LCID` switch is ignored.

The following table lists the LCIDs of the languages supported by Visual Studio.

LANGUAGE	LCID
Chinese (Simplified)	2052
Chinese (Traditional)	1028
English	1033
French	1036
German	1031
Italian	1040
Japanese	1041
Korean	1042
Spanish	3082

Example

This example loads the IDE with English resources strings.

```
devenv /LCID 1033
```

See also

- [Devenv command-line switches](#)
- [International Settings, Environment, Options Dialog Box](#)
- [Customizing window layouts](#)

/Log (devenv.exe)

10/18/2019 • 2 minutes to read • [Edit Online](#)

Logs all activity to the log file for troubleshooting. This file appears after you've called `devenv /log` at least once. By default, the log file is located here:

%APPDATA%\Microsoft\VisualStudio\<Version>\ActivityLog.xml

where <Version> is the Visual Studio version. However, you may specify a different path and file name.

Syntax

```
devenv /Log NameOfFile
```

Arguments

- *NameOfFile*

Required. The full path and name of the log file to save to.

Remarks

This switch must appear at the end of the command line, after all other switches.

The log is written only for all instances of Visual Studio that you've opened with the `/Log` switch.

Example

This example directs logging to the `MyVSLog.xml` file in the user's home directory.

```
devenv /log "%USERPROFILE%\MyVSLog.xml"
```

See also

- [Devenv command-line switches](#)

/NoSplash (devenv.exe)

2/8/2019 • 2 minutes to read • [Edit Online](#)

Prevents the splash screen from being shown.

Syntax

```
devenv /NoSplash [File1[ FileN]...]
```

Arguments

- *File1*

Optional. The file to open in an existing instance of Visual Studio. If no instance of Visual Studio exists, a new instance is created with a simplified window layout, and the tool opens *File1* in the new instance.

- *FileN*

Optional. One or more additional files to open in the existing instance of Visual Studio.

Remarks

This switch hides the splash screen. Leaving this switch out causes the splash screen to be shown. If you want to examine the splash screen further (for example, to check the VS Package product icon), use the [/Splash](#) switch.

The `/NoSplash` switch can be combined with other switches, such as [/Run](#) or [/DebugExe](#).

Example

All three of the examples open the IDE without displaying the splash screen. The second example also compiles the specified solution and runs the built executable. The third example opens the specified executable for debugging in the IDE.

```
devenv /nosplash  
devenv /nosplash /run MySolution.sln  
devenv /nosplash /debugexe MySolution.exe
```

See also

- [Devenv command-line switches](#)
- [Devenv command-line switches for VS Package development](#)

/Out (devenv.exe)

10/18/2019 • 2 minutes to read • [Edit Online](#)

Specifies a file to store and display errors when you [run](#), [run and exit](#), [upgrade](#), [build](#), [rebuild](#), [clean](#), or [deploy](#) a solution.

Syntax

```
devenv /Out FileName
```

Arguments

- *FileName*

Required. The path and name of the file to receive output when you build an executable.

Remarks

If a nonexistent file name is specified, the file is created automatically. Otherwise, the file already exists, and the results are appended to the existing contents of the file.

Command-line build errors are displayed in the **Command** window and the Solution Builder view of the **Output** window. This switch is useful for viewing results of unattended builds.

Example

This example runs `MySolution` and writes errors to the file `MyErrorLog.txt`.

```
devenv /run "%USERPROFILE%\source\repos\MySolution\MySolution.sln" /out "C:\MyErrorLog.txt"
```

See also

- [Devenv command-line switches](#)
- [/Run \(devenv.exe\)](#)
- [/RunExit \(devenv.exe\)](#)
- [/Upgrade \(devenv.exe\)](#)
- [/Clean \(devenv.exe\)](#)
- [/Build \(devenv.exe\)](#)
- [/Rebuild \(devenv.exe\)](#)
- [/Deploy \(devenv.exe\)](#)

/Project (devenv.exe)

10/18/2019 • 2 minutes to read • [Edit Online](#)

Identifies a single project within the specified solution configuration to build, clean, rebuild, or deploy.

Syntax

```
devenv SolutionName {/Build|/Clean|/Deploy|/Rebuild} [SolnConfigName [/Project ProjName [/ProjectConfig ProjConfigName]] [/Out OutputFilename]]
```

Arguments

- *SolutionName*

Required. The full path and name of the solution file.

- { /Build | /Clean | /Deploy | /Rebuild }

Required. [Builds](#), [cleans](#), [deploys](#), or [rebuilds](#) the project.

- *SolnConfigName*

Optional. The name of the solution configuration (such as `Debug` or `Release`) applied to the solution named in *SolutionName*. If more than one solution platform is available, you must also specify the platform (for example, `Debug|Win32`). If this argument is unspecified or an empty string (`""`), the tool uses the solution's active configuration.

- `/Project` *ProjName*

Optional. The path and name of a project file within the solution. You can enter the project's display name or a relative path from the *SolutionName* folder to the project file. You can also enter the full path and name of the project file.

- `/ProjectConfig` *ProjConfigName*

Optional. The project's build configuration name (such as `Debug` or `Release`) to be applied to the `/Project` named. If more than one solution platform is available, you must also specify the platform (for example, `Debug|Win32`).

- `/out` *OutputFilename*

Optional. The name of a file that you want to send the tool's output to. If the file already exists, the tool appends the output to the end of the file.

Remarks

- Must be used part of a `devenv` `/Build`, `/Clean`, `/Rebuild`, or `/Deploy` command.
- Enclose strings that include spaces in double quotation marks.
- Summary information for builds, including errors, can be displayed in the **Command** window, or in any log file specified with the `/out` switch.

Example

This example builds the project `csharpWinApp`, using the `Debug` project build configuration within `MySolution`.

```
devenv "%USERPROFILE%\source\repos\MySolution\MySolution.sln" /build Debug /project  
"CSharpWinApp\CSharpWinApp.csproj" /projectconfig Debug
```

See also

- [Devenv command-line switches](#)
- [/ProjectConfig \(devenv.exe\)](#)
- [/Build \(devenv.exe\)](#)
- [/Clean \(devenv.exe\)](#)
- [/Rebuild \(devenv.exe\)](#)
- [/Deploy \(devenv.exe\)](#)
- [/Out \(devenv.exe\)](#)

/ProjectConfig (devenv.exe)

10/18/2019 • 2 minutes to read • [Edit Online](#)

Specifies a project build configuration to be applied when you build, clean, rebuild, or deploy the project named in the `/Project` argument.

Syntax

```
devenv SolutionName {/Build|/Clean|/Deploy|/Rebuild} [SolnConfigName [/Project ProjName [/ProjectConfig ProjConfigName]] [/Out OutputFilename]]
```

Arguments

- *SolutionName*

Required. The full path and name of the solution file.

- { `/Build` | `/Clean` | `/Deploy` | `/Rebuild` }

Required. [Builds](#), [cleans](#), [deploys](#), or [rebuids](#) the project.

- *SolnConfigName*

Optional. The name of the solution configuration (such as `Debug` or `Release`) to be applied to the solution named in *SolutionName*. If more than one solution platform is available, you must also specify the platform (for example, `Debug|Win32`). If this argument is unspecified or an empty string (`""`), the tool uses the solution's active configuration.

- `/Project` *ProjName*

Optional. The path and name of a project file within the solution. You can enter the project's display name or a relative path from the *SolutionName* folder to the project file. You can also enter the full path and name of the project file.

- `/ProjectConfig` *ProjConfigName*

Optional. The project's build configuration name (such as `Debug` or `Release`) to be applied to the `/Project` named. If more than one solution platform is available, you must also specify the platform (for example, `Debug|Win32`).

- `/out` *OutputFilename*

Optional. The name of a file that you want to send the tool's output to. If the file already exists, the tool appends the output to the end of the file.

Remarks

The `/ProjectConfig` switch must be used with the `/Project` switch as part of a `/Build` , `/Clean` , `/Deploy` , or `/Rebuild` command.

Enclose strings that include spaces in double quotes.

Summary information for builds, including errors, can be displayed in the command window, or in any log file

specified with the `/out` switch.

Example

The following command builds the project `csharpWinApp`, using the `Debug` project build configuration within `MySolution`:

```
devenv "%USERPROFILE%\source\repos\MySolution\MySolution.sln" /build Debug /project  
"CSharpWinApp\CSharpWinApp.csproj" /projectconfig Debug
```

See also

- [Devenv command-line switches](#)
- [/Project \(devenv.exe\)](#)
- [/Build \(devenv.exe\)](#)
- [/Clean \(devenv.exe\)](#)
- [/Rebuild \(devenv.exe\)](#)
- [/Deploy \(devenv.exe\)](#)
- [/Out \(devenv.exe\)](#)

/Rebuild (devenv.exe)

10/18/2019 • 2 minutes to read • [Edit Online](#)

Cleans and then builds the specified solution configuration.

Syntax

```
devenv SolutionName /Rebuild [SolnConfigName [/Project ProjName [/ProjectConfig ProjConfigName]] [/Out OutputFilename]]
```

Arguments

- *SolutionName*

Required. The full path and name of the solution file.

- *SolnConfigName*

Optional. The name of the solution configuration (such as `Debug` or `Release`) to be used to rebuild the solution named in *SolutionName*. If more than one solution platform is available, you must also specify the platform (for example, `Debug|Win32`). If this argument is unspecified or an empty string (`" "`), the tool uses the solution's active configuration.

- `/Project` *ProjName*

Optional. The path and name of a project file within the solution. You can enter the project's display name or a relative path from the *SolutionName* folder to the project file. You can also enter the full path and name of the project file.

- `/ProjectConfig` *ProjConfigName*

Optional. The project's build configuration name (such as `Debug` or `Release`) to be used when rebuilding the `/Project` named. If more than one solution platform is available, you must also specify the platform (for example, `Debug|Win32`). If this switch is specified, it overrides the *SolnConfigName* argument.

- `/out` *OutputFilename*

Optional. The name of a file that you want to send the tool's output to. If the file already exists, the tool appends the output to the end of the file.

Remarks

- This switch does the same thing as the **Rebuild Solution** menu command within the IDE.
- Enclose strings that include spaces in double quotation marks.
- Summary information for cleaning and building, including errors, can be displayed in the **Command** window, or in any log file specified with the `/Out` switch.

Example

This example cleans and rebuilds the project `CSharpWinApp`, using the `Debug` project build configuration within

MySolution .

```
devenv "%USERPROFILE%\source\repos\MySolution\MySolution.sln" /rebuild Debug /project  
"CSharpWinApp\CSharpWinApp.csproj" /projectconfig Debug
```

See also

- [Devenv command-line switches](#)
- [/Build \(devenv.exe\)](#)
- [/Clean \(devenv.exe\)](#)
- [/Out \(devenv.exe\)](#)

/ResetSettings (devenv.exe)

10/18/2019 • 2 minutes to read • [Edit Online](#)

Restores Visual Studio default settings and automatically launches the Visual Studio IDE. This switch optionally resets the settings to a specified settings file.

The default settings come from the profile that was selected when Visual Studio was first launched.

TIP

To learn how to reset settings using the integrated development environment (IDE), see [Reset settings](#).

Syntax

```
devenv /ResetSettings [SettingsFile|DefaultCollectionSpecifier]
```

Arguments

- *SettingsFile*

Optional. The full path and name of the settings file to apply to Visual Studio.

- *DefaultCollectionSpecifier*

Optional. A specifier representing a default collection of settings to restore. Choose one of the default collection specifiers listed in the table.

DEFAULT COLLECTION NAME	COLLECTION SPECIFIER
General	General
JavaScript	JavaScript
Visual Basic	VB
Visual C#	CSharp
Visual C++	VC
Web Development	Web
Web Development (Code Only)	WebCode

Remarks

If no *SettingsFile* is specified, the IDE opens using the existing settings.

Example

The first example applies the settings stored in the file `MySettings.vssettings`.

The second example restores the Visual C# default profile.

```
devenv /resetsettings "%USERPROFILE%\MySettings.vssettings"  
devenv /resetsettings CSharp
```

See also

- [Environment settings](#)
- [Personalize the Visual Studio IDE](#)
- [Devenv command-line switches](#)

/Run (devenv.exe)

10/21/2019 • 2 minutes to read • [Edit Online](#)

Compiles and runs the specified project or solution.

Syntax

```
devenv {/Run|/R} {SolutionName|ProjectName} [/Out OutputFilename]
```

Arguments

- *SolutionName*

The full path and name of a solution file.

- *ProjectName*

The full path and name of a project file.

- `/out` *OutputFilename*

Optional. The name of a file that you want to send the tool's output to. If the file already exists, the tool appends the output to the end of the file.

Remarks

Compiles and runs the specified project or solution according to the settings specified for the active solution configuration. This switch launches the IDE and leaves it active after the project or solution has completed running.

- Enclose strings that include spaces in double quotation marks.
- Summary information, including errors, can be displayed in the **Command** window, or in any log file specified with the `/out` switch.

Example

This example runs the solution `MySolution` using the active deployment configuration.

```
devenv /run "%USERPROFILE%\source\repos\MySolution\MySolution.sln"
```

See also

- [Devenv command-line switches](#)
- [/Runexit \(devenv.exe\)](#)
- [/Build \(devenv.exe\)](#)
- [/Rebuild \(devenv.exe\)](#)
- [/Out \(devenv.exe\)](#)

/RunExit (devenv.exe)

10/18/2019 • 2 minutes to read • [Edit Online](#)

Compiles and runs the specified project or solution, and then closes the integrated development environment (IDE).

Syntax

```
devenv /RunExit {SolutionName|ProjectName} [/Out OutputFilename]
```

Arguments

- *SolutionName*

The full path and name of a solution file.

- *ProjectName*

The full path and name of a project file.

- `/out` *OutputFilename*

Optional. The name of a file that you want to send the tool's output to. If the file already exists, the tool appends the output to the end of the file.

Remarks

Compiles and runs the specified project or solution according to the settings specified for the active solution configuration. This switch minimizes the IDE while the project or solution is run. It closes the IDE after the project or solution has completed running.

- Enclose strings that include spaces in double quotation marks.
- Summary information, including errors, can be displayed in the **Command** window, or in any log file specified with the `/out` switch.

Example

This example runs the solution `Mysolution` in a minimized IDE using the active deployment configuration, and then closes the IDE.

```
devenv /runexit "%USERPROFILE%\source\repos\MySolution\MySolution.sln"
```

See also

- [Devenv command-line switches](#)
- [/Run \(devenv.exe\)](#)
- [/Build \(devenv.exe\)](#)
- [/Rebuild \(devenv.exe\)](#)
- [/Out \(devenv.exe\)](#)

/SafeMode (devenv.exe)

10/18/2019 • 2 minutes to read • [Edit Online](#)

Starts Visual Studio in safe mode, loading only the default environment and services.

Syntax

```
devenv /SafeMode
```

Remarks

This switch prevents all third-party VS Packages from loading when Visual Studio starts, allowing stable execution.

Example

The following example starts Visual Studio in safe mode.

```
devenv /safemode
```

See also

- [Devenv command-line switches](#)

/Upgrade (devenv.exe)

10/18/2019 • 2 minutes to read • [Edit Online](#)

Updates the solution file and all of its project files, or the project file specified, to the current Visual Studio formats for these files.

Syntax

```
devenv {SolutionFile|ProjectFile} /Upgrade [/Out OutputFilename]
```

Arguments

- *SolutionFile*

Required if you're upgrading an entire solution and its projects. The path and name of a solution file. You can enter just the name of the solution file, or a full path and the name of the solution file. If the folder or file named doesn't yet exist, it's created.

- *ProjectFile*

Required if you're upgrading a single project. The path and name of a project file within the solution. You can enter just the name of the project file, or a full path and the name of the project file. If the folder or file named doesn't yet exist, it's created.

- `/Out` *OutputFilename*

Optional. The name of a file that you want to send the tool's output to. If the file already exists, the tool appends the output to the end of the file.

Remarks

Backups are automatically created and copied to a directory named Backup that's created in the current directory.

Source-controlled solutions or projects must be checked out before they can be upgraded.

Using the `/Upgrade` switch doesn't open Visual Studio. Results of the upgrade can be seen in the Upgrade Report for the development language of the solution or project. No error or usage info is returned. For more information on upgrading projects in Visual Studio, see [Port, Migrate, and Upgrade Visual Studio Projects](#).

Example

This example upgrades a solution file named "MyProject.sln".

```
devenv "%USERPROFILE%\source\repos\MyProject\MyProject.sln" /upgrade
```

See also

- [Devenv command-line switches](#)

/UseEnv (devenv.exe)

10/18/2019 • 2 minutes to read • [Edit Online](#)

Starts Visual Studio and loads certain environment variables for compilation.

NOTE

This switch is installed with the **Desktop development with C++** workload.

Syntax

```
devenv /UseEnv {SolutionName|ProjectName}
```

Arguments

- *SolutionName*

The full path and name of a solution file.

- *ProjectName*

The full path and name of a project file.

Remarks

This switch affects the Visual Studio IDE in the project properties for **VC++ Directories**. If you specify the `/UseEnv` switch, the **VC++ Directories** node shows the values for the PATH, INCLUDE, LIBPATH, and LIB environment variables. (It also shows values for **Source Directories** and **Exclude Directories**.) Otherwise, the node replaces the environment variables with five directory values: **Executable Directories**, **Include Directories**, **Reference Directories**, **Library Directories**, and **Library WinRT Directories**.

TIP

You access the project properties by right-clicking a C++ project and selecting **Properties**. In the **Property Pages** dialog box, select **Configuration Properties** and then **VC++ Directories**.

When a project name is specified with this switch, the tool displays the environment variables for all projects within the project's parent solution.

Example

The following example starts Visual Studio and loads environment variables into the property pages of the `MySolution` solution.

```
devenv.exe /useenv "%USERPROFILE%\source\repos\MySolution\MySolution.sln"
```

See also

- [Devenv command-line switches](#)
- [VC++ Directories Property Page \(Windows\)](#)

Secure applications

10/18/2019 • 2 minutes to read • [Edit Online](#)

You should consider security in all aspects of your application development, from design to deployment. Start by running Visual Studio as securely as possible. See [User permissions](#).

To help you effectively develop secure applications, you should have a fundamental understanding of security concepts and the security features of the platforms for which you develop. You should also understand secure coding techniques.

Code for security

Most coding errors that result in security vulnerabilities occur because developers make incorrect assumptions when working with user input, or because they don't fully understand the platform for which they're developing.

- [Secure coding guidelines](#) describes the different ways .NET code can be designed to work with the security system.
- [Security best practices for C++](#) contains information about security tools and practices for C++ developers.

Build for security

Security is also an important consideration in the build process. A few additional steps can improve the security of a deployed app and help prevent unauthorized reverse engineering, spoofing, or other attacks:

- [Dotfuscator](#) is free and helps to protect .NET assemblies from reverse-engineering and unauthorized use such as unauthorized debugging.
- [Strong-name signing](#) can be used to uniquely identify software components and prevent name spoofing.

See also

- [Security in .NET](#)
- [Azure security](#)
- [Windows 10 Mobile security guide](#)
- [Apache Cordova platform security features](#)
- [ASP.NET Core security](#)
- [Windows Forms security](#)

User permissions and Visual Studio

10/18/2019 • 2 minutes to read • [Edit Online](#)

For reasons of security, you should run Visual Studio as a normal user whenever possible.

WARNING

You should also make sure not to compile, launch, or debug any Visual Studio solution that does not come from a trusted person or a trusted location.

You can do nearly everything in the Visual Studio IDE as a normal user. You need administrator permissions to complete the following tasks:

AREA	TASK	FOR MORE INFORMATION
Installation	Install Visual Studio.	Install Visual Studio
	Install, update, or remove local Help content.	Install and manage local Help content
Toolbox	Add classic COM controls to Toolbox .	Toolbox
Building	Use post-build events that register a component.	Understand custom build steps and build events
	Include a registration step when you build C++ projects.	
Debugging	Debug applications that run with elevated permissions.	Debugger settings and preparation
	Debug applications that run under a different user account, such as ASP.NET websites.	Debug ASP.NET and AJAX applications
	Debug in Zone for XAML Browser Applications (XBAP).	WPF host (PresentationHost.exe)
	Use the emulator to debug cloud service projects for Microsoft Azure.	Debug a cloud service in Visual Studio
	Configure a firewall for remote debugging.	Remote debugging
Performance tools	Attaching to an elevated application.	Beginners guide to performance profiling
	Use the GPU Profiler.	GPU profiling

AREA	TASK	FOR MORE INFORMATION
Deployment	Deploy a web application to Internet Information Services (IIS) on a local computer.	Deploy an ASP.NET web app using Visual Studio

Run Visual Studio as an administrator

If you need to run Visual Studio as an administrator, follow these steps to open the IDE:

NOTE

These instructions are for Windows 10. They are similar for other versions of Windows.

1. Open the **Start** menu, and scroll to Visual Studio 2017.
2. From the right-click or context menu of **Visual Studio 2017**, select **More > Run as administrator**.

When Visual Studio starts, **(Administrator)** appears after the product name in the title bar.

1. Open the **Start** menu, and scroll to Visual Studio 2019.
2. From the right-click or context menu of **Visual Studio 2019**, select **More > Run as administrator**.

When Visual Studio starts, **(Administrator)** appears after the product name in the title bar.

You can also modify the application shortcut to always run with administrative permissions.

See also

- [Port, migrate, and upgrade Visual Studio projects](#)
- [Install Visual Studio](#)

Configure Visual Studio as a WIP-exempt app

10/18/2019 • 2 minutes to read • [Edit Online](#)

Windows Information Protection (WIP) helps to protect enterprise data from leaking through apps like email, social media, and the public cloud, which are outside of the enterprise's control. WIP helps to protect against accidental data leakage on enterprise-owned devices and personal devices, without requiring changes to your environment or other apps.

Enlightened apps for WIP are expected to prevent enterprise data from going to unprotected network locations, and to avoid encrypting personal data. Visual Studio is not an enlightened app, so it doesn't work in WIP-enabled environments unless you exempt it. Follow the steps in this article to enable Visual Studio to function on a WIP-enabled machine.

Configure VS as a WIP-exempt app

You can exempt Visual Studio from WIP restrictions but still allow it to use enterprise data. WIP-exempt apps can connect to enterprise cloud resources using an IP address or a hostname. No encryption is applied, and the app can access local files.

To exempt Visual Studio from WIP, follow the [steps to exempt a desktop app](#).

Create a WIP-exempt AppLocker policy file

Because Visual Studio includes multiple binaries, [create an WIP-exempt AppLocker policy file](#). AppLocker allows you to automatically generate rules for all files within a folder.

Add AppCompat to the Enterprise cloud resource policy

To specify where Visual Studio can access enterprise data on your network, follow these [steps to define where your protected apps can find and send enterprise data](#). To stop Windows from blocking connections to cloud resources through an IP address, make sure to add the /*AppCompat*/ string to the setting.

See also

- [App behavior with WIP](#)

Support for bidirectional languages in Visual Studio

10/18/2019 • 2 minutes to read • [Edit Online](#)

Visual Studio can display Arabic and Hebrew text correctly and lets you enter bidirectional text for object names and values.

NOTE

In order to enter and display bidirectional languages, you must be working with a version of Windows that is configured with the appropriate language. This can either be an English version of Windows with the appropriate language pack installed, or the appropriately localized version of Windows.

Fully supported features

At design time in Visual Studio, you can use bidirectional languages while entering text, naming objects, and when saving and opening files.

Text entry

Visual Studio supports Unicode, so if your system is set to the appropriate locale and input language, you can enter text in Arabic or Hebrew. (Arabic support includes Kashida and Diacritics.)

Arabic or Hebrew object names

You can use bidirectional languages to assign names to solutions, projects, files, folders, and so on. In code, you can use bidirectional languages for the names of variables, classes, object, attributes, metadata, and other elements.

When working with Arabic, you can use any Arabic characters including Kashida and Diacritics.

The following elements can be named using Arabic or Hebrew and are handled correctly by Visual Studio:

- Solution, project, and file names, including any folders you include in the project path.

Solution Explorer displays solution and element names correctly.

- File contents.

You can open or save files with Unicode encoding or with a selected code page.

- Data elements.

Server Explorer displays these elements correctly and you can edit them.

- Elements copied to the Windows Clipboard.
- Attributes and metadata.
- Property values.

You can use Arabic or Hebrew text in the **Properties** window. The window allows you to switch between right-to-left and left-to-right reading order using standard Windows keystrokes (**Ctrl+RightShift** for right-to-left, and **Ctrl+LeftShift** for left-to-right).

- Code and literal text.

In the code editor, you can use Arabic or Hebrew to name classes, functions, variables, properties, string literals, attributes, and so on. However, the editor does not support right-to-left reading order; text always

starts at the left margin.

TIP

You should place string literals in resource files instead of hard-coding them into your programs. For more information, see [Resources in desktop apps \(.NET Framework\)](#).

NOTE

You must be consistent in how you refer to objects named in Arabic and Hebrew. For example, if you use Kashida in naming an Arabic variable, you must always use Kashida when referring to that variable or errors will result.

- Code comments. You can create comments in Arabic or Hebrew. You can also use these languages in the comment builder tool.

File encoding

You can save and open files with a language-specific or Unicode encoding. For more information, see [How to: Save and open files with encoding](#).

Right-to-left reading order

Visual Studio has limited support for right-to-left reading order. By default, text-entry controls in Visual Studio use left-to-right reading order. In most cases, you can use standard Windows gestures to switch reading order. For example, you can press **Ctrl+RightShift** to switch the **Properties** window to support right-to-left reading order for property values.

Right-to-left reading order is not supported in the following places in Visual Studio:

- Check boxes, drop-down lists, and other controls in Visual Studio dialog boxes always use left-to-right reading order.
- The code editor (and text editor) does not support right-to-left reading order. You can enter text in a bidirectional language, but the reading order is always left-to-right.

See also

- [Develop globalized and localized apps](#)

Microsoft Help Viewer

7/11/2019 • 3 minutes to read • [Edit Online](#)

You can install and view content for various products and technologies on your local computer by using Microsoft Help Viewer. These products include Visual Studio, .NET, language reference, SQL Server, and Windows Development. Help Viewer enables you to:

- Download sets of content, which are also referred to as books. This can be useful if you need to work "offline" and still have access to documentation.
- Find topics by title by browsing and searching the table of contents.
- Look up subjects in the index.
- Find information by using full-text search.
- View, bookmark, and print topics.

To install Help Viewer, see [Microsoft Help Viewer installation](#). To start reading help topics in the Help Viewer rather than online, go to the **Help** menu in Visual Studio, and then choose **Set Help Preference > Launch in Help Viewer**.

TIP

Another way to download content locally so you can view it when you don't have an internet connection is to download a PDF version of it. Many documentation sets on docs.microsoft.com include a link at the bottom of the table of contents (TOC) to download a PDF file that contains all the articles for that TOC.

[↓ Download PDF](#)

Help Viewer tour

You can find information in installed content by using the navigation tabs, view installed content in the topic tab or tabs, and manage content by using the **Manage Content** tab. You can also perform additional tasks by using the buttons on the toolbar and find additional information in the lower-right corner of the window.

Navigation tabs

TAB	DESCRIPTION
Contents	Displays installed content as a hierarchy (table of contents). You can specify criteria to filter the titles that appear.
Index	Displays an alphabetical list of indexed terms. You can search the index, specify criteria to filter the entries, and require that index entries either contain or start with text that you specify.
Favorites	You can "favorite" topics by choosing the Add to Favorites button, and the topics appear in this tab. The History section displays a list of topics that you've viewed recently.

TAB	DESCRIPTION
Search	Provides a text box where you can search for terms anywhere in the content, including code and topic titles.

View topics

Each topic appears in its own tab, and you can open multiple topics at the same time.

Manage content

You can install, update, move, and delete content by using the **Manage Content** tab. At the top of the tab, you can use the **Installation source** control to specify whether to install books from a network location or from a disk or URI. The **Local store path** box shows where books are installed on the local computer, and you can move them to a different location by choosing the **Move** button.

The content list shows which books you can install or have already installed, whether an update is available, and how large each book is. You can install or remove one or more books by choosing the appropriate **Add** or **Remove** links and then choosing the **Update** button under the **Pending changes** pane. If updates are available for any books that you've already installed, you can refresh that content by choosing the **Click here to download now** link at the bottom of the window. In addition, all installed books are refreshed if updates are available when you install additional books.

NOTE

The functionality of the **Manage Content** tab may differ if the Help Viewer administrator deactivates these features, or if no internet access is available.

Toolbar buttons

The toolbar in the **Help Viewer** window contains the following buttons:

- The **Show Topic in Contents** button shows the location of the topic in the **Contents** tab.
- The **Add to Favorites** button adds the active topic to the **Favorites** tab.
- The **Find in Topic** button highlights search text in the active topic.
- The **Print** button prints or shows a preview of the active topic.
- The **Viewer Options** button displays settings such as how large the text appears, how many search results to return, how many topics to show in history, and whether to check for updates online.
- The **Manage Content** button makes the **Manage Content** tab active.
- The small triangle on the right-hand side opens a list of tabs, including topic tabs and the **Manage Content** tab. You can choose a tab name to make it the active tab.

See also

- [Microsoft Help Viewer installation](#)
- [Help Viewer administrator guide](#)
- [Install and manage local content](#)

Install and manage local content

7/11/2019 • 3 minutes to read • [Edit Online](#)

By using the Microsoft Help Viewer, you can add, remove, update, and move the Help content that is installed on your computer to fit your software development needs.

To manage content on your local computer, you must log on with an account that has administrator permissions. In addition, you might not be able to manage local content if you work in an enterprise environment, because system administrators might make those decisions for your organization. For more information, see the [Help Viewer administrator guide](#).

Change the content installation source

By default, Help Viewer installs content by using a Microsoft online service as the source. You generally shouldn't change your content source unless you work in an enterprise environment for which a system administrator has already installed content in another location.

To change the content installation source

1. On the **Manage Content** tab, choose the **Disk** option button.

NOTE

The **Disk** option isn't available if your administrator has prevented you from modifying the content installation source. For more information, see the [Help Viewer administrator guide](#).

2. Perform one of the following steps:

- Enter the path of an *.msha* file or the URL of a service endpoint.
- Choose the **Browse (...)** button to navigate to an *.msha* file.
- In the list, choose the entry that was used most recently.

Download and install content locally

If you download and install content on your local computer, you can view topics when you don't have an internet connection.

IMPORTANT

To install content, you must log on with an account that has administrative permissions.

NOTE

If the Visual Studio IDE is set to a language other than English, you can install English content, localized content, or both. However, no content appears if you install only the English version and the **Include English content in all navigation tabs and F1 requests** check box in the **Viewer Options** dialog box is cleared.

To download and install content

1. Choose the **Manage Content** tab.

2. In the content list, choose the **Add** link next to the book or books that you want to download and install.

The book is added to the **Pending changes** list, and the estimated size of the book or books that you specified appears below that list. Because some books share topics, the total download size of multiple books might be smaller than the result of adding together the sizes of every book that you specified.

3. Choose the **Update** button.

The book or books that you specified are installed along with any updates for books that you already have on your computer. Installation times vary, but you can view the progress in the status bar.

Remove local content

You can save disk space by removing unwanted content from your computer.

IMPORTANT

You must have administrative permissions to remove content.

NOTE

No content appears if the Visual Studio IDE is set to a language other than English, you remove localized content, and the **Include English content in all navigation tab and F1 requests** check box in the **Viewer Options** dialog box is cleared.

To remove content

1. Choose the **Manage Content** tab.

2. In the content list, choose the **Remove** link next to the book or books that you want to remove.

The book is added to the **Pending changes** list.

3. Choose the **Update** button.

The book or books that you specified are removed from your computer.

Update local content

The status bar indicates when updates to your installed content are available.

IMPORTANT

If you want the **Help Viewer** to automatically check for online updates, you must open the **Viewer Options** dialog box and then select the **Go online to check for content updates** check box.

To update local content

- In the lower-right corner of the status bar, choose the **Click here to download now** link.

Update times can vary, but you can view the update progress in the status bar.

Move local content

You can save disk space by moving installed content from your local computer to a network share or to another partition on your local computer.

IMPORTANT

To move content, you must log on with an account that has administrative permissions.

To move local content

1. On the **Manage Content** tab, choose the **Move** button under **Local Store Path**.

The **Move Content** dialog box opens.

2. In the **To** text box, enter a different location for the content, and then choose the **OK** button.
3. Choose the **Close** button when the content has been moved.

See also

- [Microsoft Help Viewer](#)

Find topics by using the Help Viewer index

7/11/2019 • 2 minutes to read • [Edit Online](#)

The index contains a list of keywords that are associated with topics in the installed content. Each topic might have more than one keyword associated with it, and each keyword might be associated with more than one topic. Use this index in the same way as you would use an index in a book.

To find a topic by using the index

On the **Index** tab, perform either of the following tasks:

- Specify the keyword to search for in the text box. For example, specify "update" to find topics with keywords such as "update," "updated," and "updating."

By choosing the filter button near the top of the tab, you can display either all entries that contain the text that you specify or only those entries that start with the text that you specify.

NOTE

When the filter button appears on a darker background with a border, entries must *contain* the text that you specify. If the background and border don't appear, entries must *start with* the text that you specify.

- Scroll through the index, and choose a keyword.

If the keyword that you specify is associated with only one topic, it appears. Otherwise, a list of all topics that are associated with that keyword appears.

Index search tips

Using the index is a straight-forward process; however, understanding how to best enter keywords can make your index searches more productive.

General guidelines

- Scroll through the index entries. Not all topics are indexed the same way, and the one that could most help you might be higher or lower in the list than you expected.
- Omit articles such as "an" or "the" because the index ignores them.
- Reverse the words you enter if you do not find the entries you expect.

For example, if "debugging inline assembly code" did not display any relevant entries, try typing, "assembly code, debugging inline".

- Use filters with the **Index** tab to decrease the number of results.

Syntax tips

If you do not find an entry for the word or phrase you entered, try the following:

- Type the first few letters, or root, of the word. By entering a partial string, you can get to topics that have been indexed with keywords that are singular or plural.

For example, enter "proper" to start your search above properties and property.

- Enter gerund (-ing) forms of the verb for the task you want to complete. To find more specific index entries,

append a word that describes exactly what you want.

For example, type "running" to get more entries or "running programs" to get fewer.

- Enter standalone adjectives. To narrow the results, append a word that describes exactly what you want.

For example, enter "COM+" to get a wide range of entries or "COM+ components" to get fewer.

- Enter a synonym of the word or verb you are looking for.

For example, if you entered the term "building", try "creating" instead.

See also

- [How to: Find topics in the TOC](#)
- [How to: Search for topics](#)
- [Microsoft Help Viewer](#)

How to: Find topics in the table of contents

7/11/2019 • 2 minutes to read • [Edit Online](#)

In the **Contents** tab, you can use the table of contents (TOC) to find information. The table of contents is an expandable list that contains all of the topics in the installed books. For accessibility information about how to navigate through the TOC, see [Shortcut keys \(Help Viewer\)](#).

IMPORTANT

The scope of topics available in the TOC depends on the filter you have selected.

Filter the TOC

You can filter the TOC to narrow the scope of topics that appear in the **Contents** tab. Titles appear in the list only if they contain the root of the term that you specify. For example, if you specify "troubleshooting" as a filter, only titles that contain "troubleshoot" or "troubleshooting" appear. Nodes whose titles don't contain the term are collapsed to a single node with an ellipsis (...).

1. Choose the **Contents** tab.
2. In the **Filter Contents** text box, enter a term.

NOTE

If the filter takes a long time to run, you might display results more quickly by using the `title:` advanced search operator.

Synchronize a topic with the TOC

If you have opened a topic using the index or full-text search features, you can determine where this topic is in the TOC by synchronizing the TOC with the topic window.

1. View a topic.
2. Click the **Show Topic in Contents** button on the toolbar, or press **Ctrl+S**.

The **Contents** tab opens and displays the topic's location in the TOC.

See also

- [How to: Find topics in the index](#)
- [How to: Search for topics](#)
- [Microsoft Help Viewer](#)

How to: Search for topics

7/11/2019 • 3 minutes to read • [Edit Online](#)

You can use the full-text search feature to locate all topics that contain a particular word. You can also refine and customize your search by using wildcard expressions, logical operators, and advanced search operators.

To open the **Search** tab, choose the **Search** tab in the **Help Viewer** window, or if you are a keyboard user, choose **Ctrl+E**.

To perform a full-text search

1. In the search box, type the word that you want to find.
2. In the search query, specify which logical or advanced search operators to apply to the search, if any. To search all available help, don't use operators.

NOTE

In the **Viewer Options** dialog box, you can specify additional preferences such as the maximum number of search results to display at a time and whether to include English content if your primary locale is not English.

3. Choose the **Enter** key.

A search returns a maximum of 200 hits, by default, and displays them in the search results area. Additional version information for each result may appear, depending on the content.

4. To view a topic, choose its title from the results list.

Full-text search tips

You can create more targeted searches that return only those topics that interest you, if you understand how syntax affects your query. The syntax includes special characters, reserved words, and filters. This topic provides tips, procedures, and detailed syntax information to help you better craft your queries.

General guidelines

The following table includes some basic rules and guidelines for developing search queries in help.

SYNTAX	DESCRIPTION
Case sensitivity	Searches aren't case-sensitive. Develop your search criteria using uppercase or lowercase characters. For example, "OLE" and "ole" return the same results.
Character combinations	You can't search only for individual letters (a-z) or numbers (0-9). If you try to search for certain reserved words, such as "and", "from", and "with", they will be ignored. For more information, see Words ignored in searches later in this topic.
Evaluation order	Search queries are evaluated from left to right.

Search syntax

If you specify a search string that includes multiple words, such as "word1 word2," that string is equivalent to

typing "word1 AND word2", which returns only topics that contain all of the individual words in the search string.

IMPORTANT

- Phrase searches are not supported. If you specify more than one word in a search string, returned topics will contain all of the words that you specified but not necessarily the exact phrase that you specified.
- Use logical operators to specify the relationship between words in your search phrase. You can include logical operators, such as AND, OR, NOT, and NEAR, to further refine your search. For example, if you search for "declaring NEAR union", search results will include topics that contain the words "declaring" and "union" no more than a few words apart from each other. For more information, see [Logical operators in search expressions](#).

Filters

You can further restrict search results by using advanced search operators. Help includes three categories that you can use to filter results of a full-text search: Title, Code, and Keyword.

Ranking of search results

The search algorithm applies certain criteria to help rank search results higher or lower in the results list. In general:

1. Content that includes search words in the title is ranked higher than content that doesn't.
2. Content that includes search words in close proximity is ranked higher than content that doesn't.
3. Content that contains a higher density of the search words is ranked higher than content that has a lower density of the search words.

[Words ignored in searches \(stop words\)](#)

Commonly occurring words or numbers, which are sometimes called stop words, are automatically ignored during a full-text search. For example, if you search for the phrase "pass through", search results will display topics that contain the word "pass" but not "through".

See also

- [Logical and advanced operators](#)
- [How to: Find topics in the index](#)
- [How to: Find topics in the TOC](#)
- [Microsoft Help Viewer](#)

Logical and advanced operators in search expressions

7/11/2019 • 2 minutes to read • [Edit Online](#)

You can use logical operators and advanced search operators to refine your search of the Help content in **Help Viewer**.

Logical operators

Logical operators specify how multiple search terms should be combined in a search query. The following table shows the logical operators AND, OR, NOT and NEAR.

TO SEARCH FOR	USE	EXAMPLE	RESULT
Both terms in the same article	AND	dib AND palette	Topics that contain both "dib" and "palette".
Either term in an article	OR	raster OR vector	Topics that contain either "raster" or "vector".
First term without the second term in the same article	NOT	"operating system" NOT DOS	Topics that contain "operating system" but not "DOS".
Both terms, close together in an article	NEAR	user NEAR kernel	Topics that contain "user" within close proximity of "kernel".

IMPORTANT

You must enter logical operators in all capital letters for the search engine to recognize them.

Advanced operators

Advanced search operators refine your search for content by specifying where in an article to look for the search term. The following table describes the four available advanced search operators.

TO SEARCH FOR	USE	EXAMPLE	RESULT
A term in the title of the article	<code>title:</code>	<code>title:binaryreader</code>	Topics that contain "binaryreader" in their titles.
A term in a code example	<code>code:</code>	<code>code:readdouble</code>	Topics that contain "readdouble" in a code example.
A term in an example of a specific programming language	<code>code:vb:</code>	<code>code:vb:string</code>	Topics that contain "string" in a Visual Basic code example.

TO SEARCH FOR	USE	EXAMPLE	RESULT
An article that is associated with a specific index keyword	keyword:	keyword:readbyte	Topics that are associated with the "readbyte" index keyword.

IMPORTANT

You must enter advanced search operators with a final colon and no intervening space before the colon for the search engine to recognize them.

Programming languages for code examples

You can use the `code:` operator to find content about any of several programming languages. To return examples for a specific programming language, use one of the following programming language values:

PROGRAMMING LANGUAGE	SEARCH OPERATOR SYNTAX
Visual Basic	code:vb code:visualbasic
C#	code:c# code:csharp
C++	code:cpp code:c++ code:cplusplus
F#	code:f# code:fsharp
JavaScript	code:javascript code:js
XAML	code:xaml

NOTE

The `code:` operator only finds content that is marked up with a programming language label, as opposed to content that is generically marked up as code.

See also

- [How to: Search for topics](#)
- [Microsoft Help Viewer](#)

Customize the help viewer

7/11/2019 • 2 minutes to read • [Edit Online](#)

You can customize the layout of the Help Viewer windows, as well as other options such as font size, maximum number of results, and whether to include English content.

Customizing window layout

You can customize the window layout of the Help Viewer. To restore the Help Viewer window to its default layout, open the **Viewer Options** dialog box, and then choose the **Reset** button.

Docking tabs

The Help Viewer supports standard docking functionality. By default, all tabs in the Help Viewer are docked, but you can move them, resize them, dock them in other locations, and "float" them so that they appear as independent child windows.

Opening a topic in a new tab

Choose the topic in any navigation tab, and then press **Ctrl+Enter**.

Minimize a navigation tab

Create more space for viewing topics by choosing the pin icon for the navigation tabs. When these tabs are minimized, only their labels appear on the closest edge of the window. To restore the tabs, choose the label of any tab, and then choose the pin icon again.

Changing settings in Viewer Options

You open the **Viewer Options** dialog box by choosing the **Viewer Options** button on the toolbar.

TO PERFORM THIS TASK:	TAKE THIS STEP:
Change the size of the font in which text appears	Choose a size in the Text Size list.
Change the maximum number of search results that appear in the Search tab	Choose a value in the Maximum Search Results list.
Change the maximum number of history entries that appear in the Favorites window	Choose a value in the Maximum History entries saved list.
Include or exclude English content when you view content for a non-English version of a product.	Select or clear the Include English content in all navigation tabs and F1 requests check box. Caution: This feature also controls whether you can download English content in the Manage Content tab.

See also

- [Microsoft Help Viewer](#)

Accessibility features of the Help Viewer

7/11/2019 • 2 minutes to read • [Edit Online](#)

Microsoft is committed to making its products and services easier for everyone to use. This topic includes information about the features, products, and services that help make Microsoft Help Viewer accessible for people with a wide range of abilities.

Keyboard access

You can access all features of the Help Viewer by using the keyboard. For more information, see [Shortcut Keys \(Help Viewer\)](#).

Font size

You can modify the font size in which topic text appears in the document window. For more information, see [Customize the Help Viewer](#).

Window size

You can change the width of the navigation or document windows by pointing to the divider between the two windows. When the cursor changes to a double-headed arrow, use the primary mouse button to drag the divider to the right or left.

Help Viewer position

You can reposition the Help Viewer by dragging its title bar to a different position.

See also

- [Microsoft Help Viewer](#)
- [Shortcut Keys \(Help Viewer\)](#)

Shortcut keys in Help Viewer

7/11/2019 • 5 minutes to read • [Edit Online](#)

You can navigate in the **Microsoft Help Viewer** by using the shortcut keys in the following table:

AREA	KEYSTROKE	ACTION
General Application	Space	Use instead of Enter anywhere except in edit fields.
General Application	F1	Open Help about current UI element.
General Application	F11	Toggle between full-screen view and regular view.
Toolbar	Backspace -OR- Alt+ Left Arrow	Display the previous page.
Toolbar	Alt+ Right Arrow	Display the next page.
Toolbar	Alt+ Home	Display the Help Reviewer home page.
Toolbar	Ctrl+S	Highlight the current topic in the table of contents (on the Contents tab).
Toolbar	Ctrl+D	Add the current topic to the Favorites tab.
Toolbar	Ctrl+F	Display the Find bar in the topic area so that you can search for text within the current topic.
Toolbar	Ctrl+P	Print the current page.
Toolbar	Ctrl+F2	Display a print preview of the current page.
Toolbar	Ctrl+O	Display the Viewer Options dialog box.
Toolbar	Ctrl+Shift+M	Display the Manage Contents tab.
Navigators	Alt+C -OR- Ctrl+Shift+C	Display the Contents tab.

AREA	KEYSTROKE	ACTION
Navigators	Alt+I -OR- Ctrl+Shift+I	Display the Index tab.
Navigators	Alt+F -OR- Ctrl+Shift+F	Display the Favorites tab.
Navigators	Alt+S -OR- Ctrl + E -OR- Ctrl+Shift+S	Display the Search tab.
Navigators	Alt+M -OR- Ctrl+Shift+M	Display the Manage Content tab.
Topic	Shortcut Menu key OR Shift+F10	Display the shortcut menu for the current UI element.
Topic	Up Arrow	Scroll toward the start of the document one line at a time.
Topic	Down Arrow	Scroll toward the end of the document one line at a time.
Topic	Page Up	Scroll toward the start of the document one screen at a time.
Topic	Page Down	Scroll toward the end of the document one screen at a time.
Topic	Home	Move to the start of the document.
Topic	End	Move to the end of the document.
Topic	Ctrl+F	Find search text on this page.
Topic	F5	Refresh the current page.
Topic	Ctrl+P	Print the current page.

AREA	KEYSTROKE	ACTION
Topic	Ctrl+F2	Display a print preview of the current page.
Topic	F4	Display the Properties dialog box for the current page.
Topic	Ctrl+T	Open another content tab in the foreground.
Topic	Ctrl+Click	Open a link on a new tab in the foreground.
Topic	Ctrl+Tab	Switch among tabs from left to right.
Topic	Ctrl+Shift+Tab	Switch among tabs from right to left.
Topic	Ctrl+W	Close the current tab.
Topic	Ctrl+Number	Switch to a specific tab where Number is between 1 and 9 and indicates which tab in sequence.
Topic	Ctrl+Alt+F4	Close other content tabs.
Topic	Ctrl+Shift+Plus Sign	Increase zoom by 10%.
Topic	Ctrl+Minus Sign	Decrease zoom by 10%.
Topic	Ctrl+0 (zero)	Change zoom to 100%.
Index	Tab	Shift focus from keyword entry to keyword list.
Index	Ctrl+K	Switch between showing entries that contain the keyword that you specify and entries that start with the keyword that you specify.
Favorites	Ctrl+Shift+Del	Clear your browsing history.
Favorites	Del	Delete the specified item.
Favorites	Ctrl+N	Create a folder within Favorites .
Favorites	F2	Rename the specified favorite or folder.
Contents & Index & Search	Ctrl+D	Add the specified topic to the Favorites tab.
Contents & Index & Search & Favorites	Ctrl+P	Print the specified topic.

AREA	KEYSTROKE	ACTION
Contents & Index & Search & Favorites	Ctrl+F2	Display a print preview of the specified topic.
Contents & Index & Search & Favorites	Ctrl+Click	Open the topic in a new tab.
Search	Esc	Clear the search text box.
Viewer Options	Alt+T	Change focus to the Text Size list.
Viewer Options	Alt+S	Change focus to the Maximum Search Results list.
Viewer Options	Alt+H	Change focus to the Maximum History entries saved list.
Viewer Options	Alt+E	Select or clear the Include English content in all navigation tabs and F1 requests check box if it is enabled.
Viewer Options	Alt+O	Select or clear the Go online to check for content updates check box.
Find	Enter	Change focus to the next item.
Find	Shift+Enter	Change focus to the previous item.
Find	Esc	Hides the Find text box.
Status bar	Alt+E	Open the Error dialog box if the status bar shows that an error has occurred.
Status bar	Alt+U	Download content if the status bar shows that updates are available

Window management

Keystroke	Action
Ctrl+L	Reset the Help Viewer layout to the default layout, and close all topic tabs.
Ctrl+Tab	The first keystroke gives focus to the Tab Selection menu. The next keystroke gives focus to the top menu item, and subsequent keystrokes give focus to the menu items in sequence from top to bottom. When a menu item has focus, the Enter key makes that item the active tab.
Ctrl+Shift+Tab	The first keystroke gives focus to the Tab Selection menu. The next keystroke gives focus to the bottom menu item, and subsequent keystrokes give focus to the menu items in sequence from bottom to top. When a menu item has focus, the Enter key makes that item the active tab.

Alt+I, Alt+S, Alt+C, Alt+F, Alt+M	These shortcut keys don't work when the navigation and content-management tabs are undocked.
--	--

Manage content

Keystroke	Action
Alt+D	Change the installation source to disk.
Alt+O	Change the installation source to online.
Tab	Change focus to the Local store path text box.
Tab	Change focus to the Move button.
Alt+V	Open the Move Content dialog box.
Ctrl+Alt+F	Change focus to the Filter Documentation text box.
Tab	Change focus to the documentation list.
Up Arrow and Down Arrow	Scroll through the documentation list.
Space	Add an item to the Pending changes list.
Tab	Change focus to the Pending changes list.
Up Arrow and Down Arrow	Scroll through the Pending changes list.
Space	Remove an item from the Pending changes list.
Alt+T	Apply all pending changes.

See also

- [Accessibility features of the Help Viewer](#)

The Visual Studio image library

10/24/2019 • 2 minutes to read • [Edit Online](#)

The Visual Studio Image Library contains application images that appear in Microsoft Visual Studio, Microsoft Windows, the Office system and other Microsoft software. This set of over 1,000 images can be used to create applications that look visually consistent with Microsoft software.

[Download the Visual Studio image library](#)

The image library is divided into five categories: Common Elements, Actions, Annotations, Icons, and Objects. Readme files are included in the PDF format for the Common Elements and Icon types. They contain information about how to use these images appropriately in your applications.

See also

- [Install Visual Studio](#)
- [Images, bitmaps, and metafiles](#)

Dotfuscator Community

10/18/2019 • 3 minutes to read • [Edit Online](#)

PreEmptive Protection - Dotfuscator provides comprehensive .NET application protection that easily fits into your secure software development lifecycle. Use it to harden, protect, and prune desktop, mobile, server, and embedded applications to help secure trade secrets and other intellectual property (IP), reduce piracy and counterfeiting, and protect against tampering and unauthorized debugging. Dotfuscator works on compiled assemblies without the need for additional programming or even access to source code.

PreEmptive Protection



Why protection matters

It's important to **protect your intellectual property** (IP). Your application's code contains design and implementation details, which can be considered IP. However, applications built on the .NET Framework **contain significant metadata and high-level intermediate code**, making them easy to reverse engineer, just by using one of many free, automated tools. By disrupting and stopping reverse-engineering, you can prevent unauthorized IP disclosure, as well as demonstrate that your code contains trade secrets. Dotfuscator can **obfuscate** your .NET assemblies to hinder reverse-engineering, while maintaining original application behavior.

It's also important to **protect the integrity of your application**. In addition to reverse-engineering, bad actors may attempt to pirate your application, alter the application's behavior at run time, or manipulate data. Dotfuscator can inject your application with the capability to **detect and respond to unauthorized uses**, including tampering, third-party debugging, and rooted devices.

For more information on how Dotfuscator fits into a secure software development lifecycle, see PreEmptive Solutions' [SDL App Protection page](#).

About Dotfuscator Community

Your copy of Microsoft Visual Studio includes a copy of **PreEmptive Protection - Dotfuscator Community**, free for personal use. (This free version was previously known as Dotfuscator Community Edition or Dotfuscator CE.) For instructions on how to install the version of Dotfuscator Community included with Visual Studio, see the [Installation page](#).

Dotfuscator Community offers a range of **software protection and hardening** services for developers, architects, and testers. Examples of **.NET Obfuscation** and other **Application Protection** features included in Dotfuscator Community are:

- **Renaming** of identifiers to make reverse-engineering of the compiled assemblies more difficult.
- **Anti-tamper** to detect the execution of tampered applications and terminate or respond to tampered sessions.
- **Anti-debug** to detect the attachment of a debugger to a running application and terminate or respond to debugged sessions.

- [Anti-rooted device](#) to detect if the application is running on a rooted Android device and terminate or respond to sessions on these devices.
- [Application expiration behaviors](#) that encode an "end-of-life" date and terminate expired application sessions.

For details on these features, including how they fit into your application protection strategy, see the [Capabilities page](#).

Dotfuscator Community offers basic protection out-of-the-box. Even more application protection measures are available to registered users of Dotfuscator Community, and to users of **PreEmptive Protection - Dotfuscator Professional**, the world's leading [.NET Obfuscator](#). For information about enhancing Dotfuscator, see the [Upgrades page](#).

Getting started

To begin using Dotfuscator Community from Visual Studio, type `dotfuscator` into the **Search Box** (Ctrl+Q).

- If Dotfuscator Community is already installed, **Search Box** will show the option to start Dotfuscator Community under the *Menus* heading. For details, see the [Getting Started page of the full Dotfuscator Community User Guide](#).
- If Dotfuscator Community is not yet installed, **Search Box** will instead show **Install PreEmptive Protection - Dotfuscator** under the *Individual Components* heading. See the [Installation page](#) for details.

To begin using Dotfuscator Community from Visual Studio, type `dotfuscator` into the **Quick Launch** (Ctrl+Q) search bar.

- If Dotfuscator Community is already installed, **Quick Launch** brings up the *Menu* option to start the Dotfuscator Community user interface. For details, see the [Getting Started page of the full Dotfuscator Community User Guide](#).
- If Dotfuscator Community is not yet installed, **Quick Launch** brings up the relevant *Install* option. See the [Installation page](#) for details.

You can also get the **latest version** of Dotfuscator Community from [the Dotfuscator Downloads page on preemptive.com](#).

Full documentation

This page and its subpages provide a high-level overview of Dotfuscator Community's features, as well as instructions for installing the tool.

See [the full Dotfuscator Community User Guide at preemptive.com](#) for detailed usage instructions, including how to start using the Dotfuscator Community user interface.

Capabilities of Dotfuscator

10/21/2019 • 2 minutes to read • [Edit Online](#)

This page focuses on the capabilities of Dotfuscator Community with some references to advanced options available through [upgrades](#).

Dotfuscator Community is a *post-build* system for .NET applications. With it, Visual Studio users are able to [obfuscate assemblies](#) and inject [active defense measures](#) into the application - all without Dotfuscator needing to access the original source code. Dotfuscator protects your application in multiple ways, creating a layered protection strategy.

Dotfuscator Community supports a wide range of .NET assembly and application types, including [Universal Windows Platform \(UWP\)](#) and [Xamarin](#).

Intellectual Property Protection

Your application's design, behavior, and implementation are forms of intellectual property (IP). However, applications created for .NET are essentially open books; it's easy to reverse engineer .NET assemblies, [as they contain high-level metadata and intermediate code](#).

Dotfuscator Community includes basic [.NET obfuscation](#) in the form of [renaming](#). Obfuscating your code with Dotfuscator reduces the risk of unauthorized access to source code through reverse engineering, as important naming information will no longer be public. Obfuscation also shows effort on your part to protect your code from examination - a valuable step in establishing that your IP is legally protected as trade secret.

Many of the [application integrity protection features](#) of Dotfuscator Community further hinder reverse engineering. For instance, a bad actor may attempt to attach a debugger to a running instance of your application in order to understand the program logic. Dotfuscator can inject [anti-debug behavior](#) into your application to obstruct this.

Application Integrity Protection

In addition to protecting your source code, it's also important to ensure your application is used as designed. Attackers can attempt to hijack your application in order to circumvent licensing policies (that is, software piracy), to steal or manipulate sensitive data handled by the application, or to change the behavior of the application.

Dotfuscator Community can inject [application validation code](#) into your assemblies, including [anti-tamper](#), [anti-debug](#), and [anti-rooted device](#) measures. When an invalid application state is detected, the validation code can [call upon application code to address the situation in an appropriate way](#). Or, if you prefer not to write code to handle invalid uses of the application, Dotfuscator can also inject [response](#) behaviors, without requiring any modification to your source code.

Many of these same methods may also be used to enforce [end-of-life deadlines](#) for evaluation or trial software.

See also

[This topic in the full Dotfuscator Community User Guide](#)

Install Dotfuscator Community

10/18/2019 • 2 minutes to read • [Edit Online](#)

Dotfuscator Community is an optional component of Visual Studio. These instructions explain how to install it.

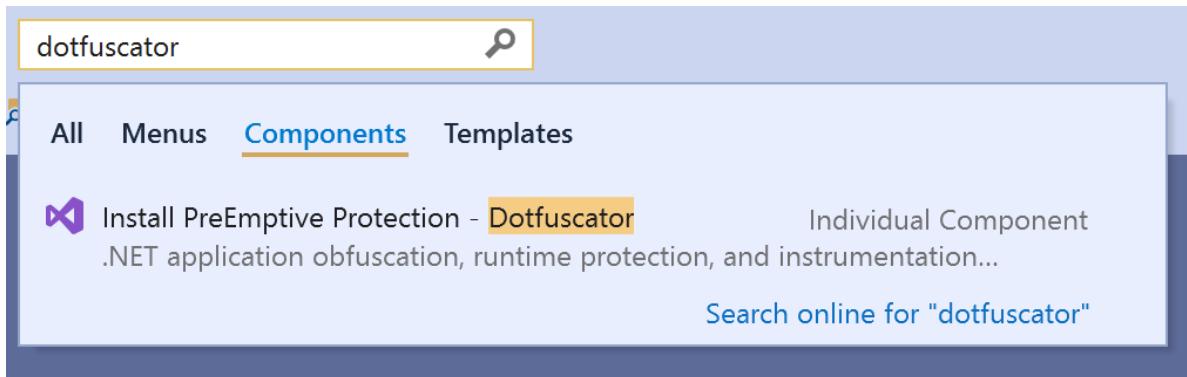
NOTE

In addition to the versions of Dotfuscator Community shipped with releases of Visual Studio, PreEmptive Solutions also periodically provides updated versions on its website. If you want to download the **latest version** directly instead of installing from Visual Studio, [click here to go to the Dotfuscator Downloads page](#).

Within Visual Studio

You can install Dotfuscator Community from the Visual Studio IDE:

1. In the **Search Box** (Ctrl+Q), type `dotfuscator`.



2. In the search results shown, under the *Components* heading, select **Install PreEmptive Protection - Dotfuscator**.

- If you instead see, under the *Menus* heading, **PreEmptive Protection - Dotfuscator Community**, then Dotfuscator Community is already installed. Select that option to [get started](#).

3. A Visual Studio Installer window will launch, pre-configured to install Dotfuscator Community.

NOTE

You may be required to provide administrator credentials to continue.

4. In the Visual Studio Installer window, click *Install*.

We recommend that you install the following:

- PreEmptive Protection - Dotfuscator

Total space required 51 MB

[View full installation details](#)

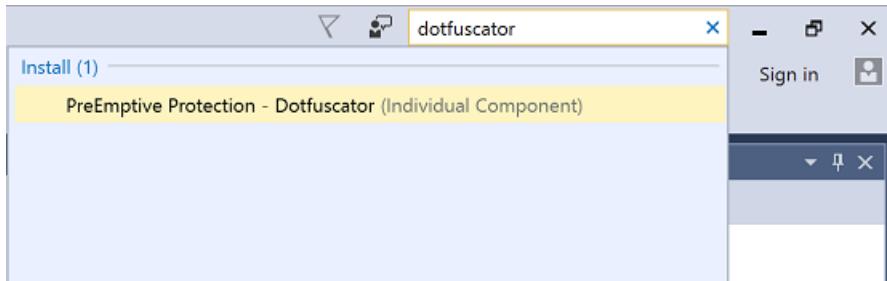
By continuing, you agree to the [license](#) for the Visual Studio edition you selected. We also offer the ability to download other software with Visual Studio. This software is licensed separately, as set out in the [3rd Party Notices](#) or in its accompanying license. By continuing, you also agree to those licenses.

 Visual Studio will close and reopen.

[Install](#)

You can install Dotfuscator Community from the Visual Studio IDE:

1. In the **Quick Launch** (Ctrl+Q) search bar, type `dotfuscator`.



2. In the Quick Launch results shown, under the *Install* heading, select **PreEmptive Protection - Dotfuscator (Individual Component)**.

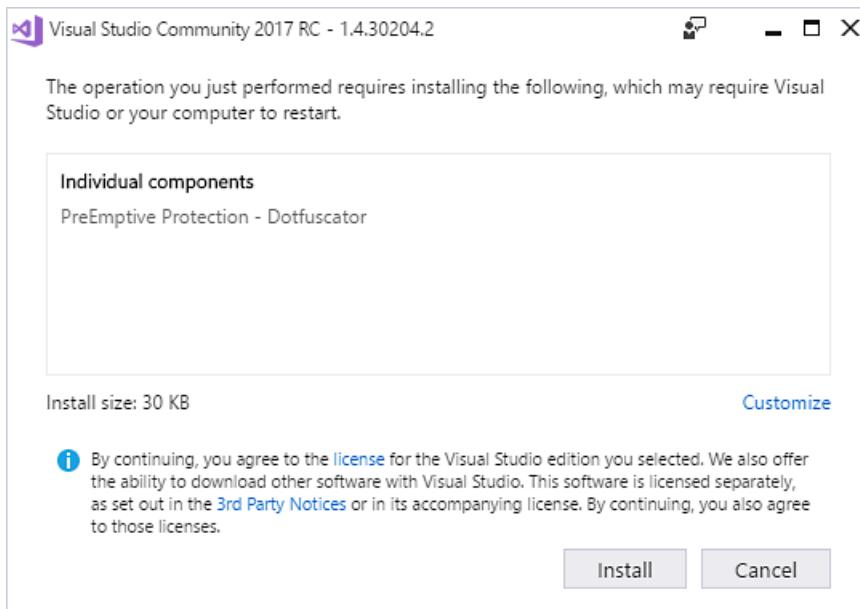
- If you instead see, under the *Menus* heading, **Tools - PreEmptive Protection - Dotfuscator**, then Dotfuscator CE is already installed. Select that option to [get started](#).

3. A Visual Studio Installer window will launch, pre-configured to install Dotfuscator CE.

NOTE

You may be required to provide administrator credentials to continue.

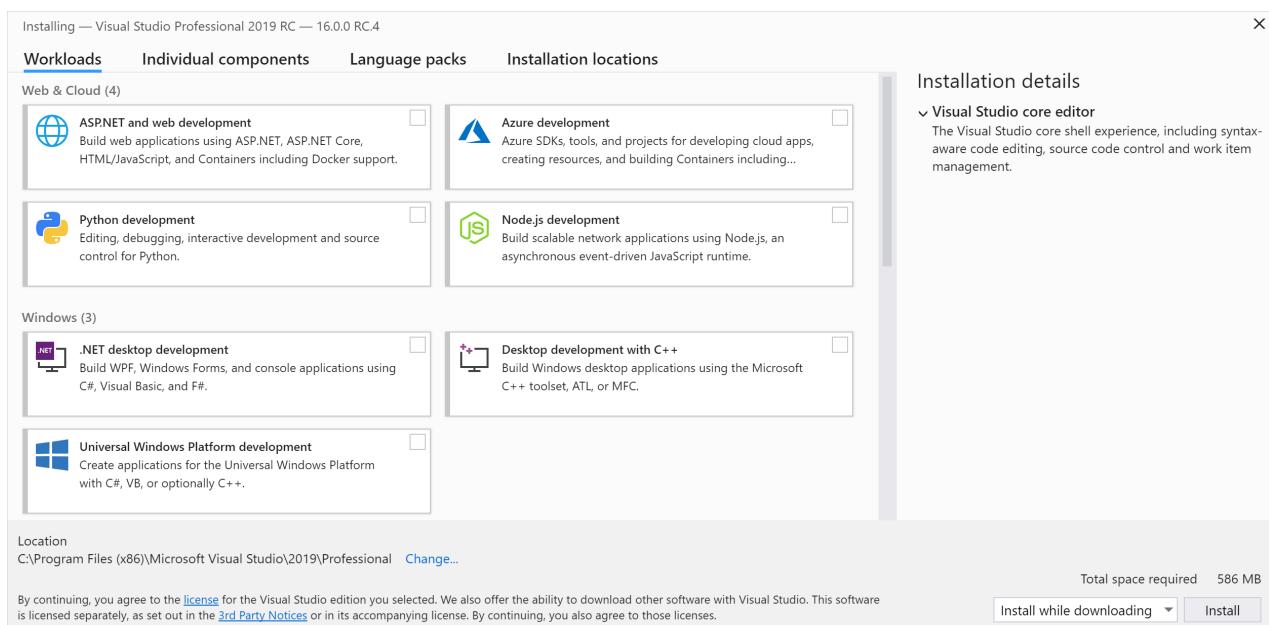
4. In the Visual Studio Installer window, click *Install*.

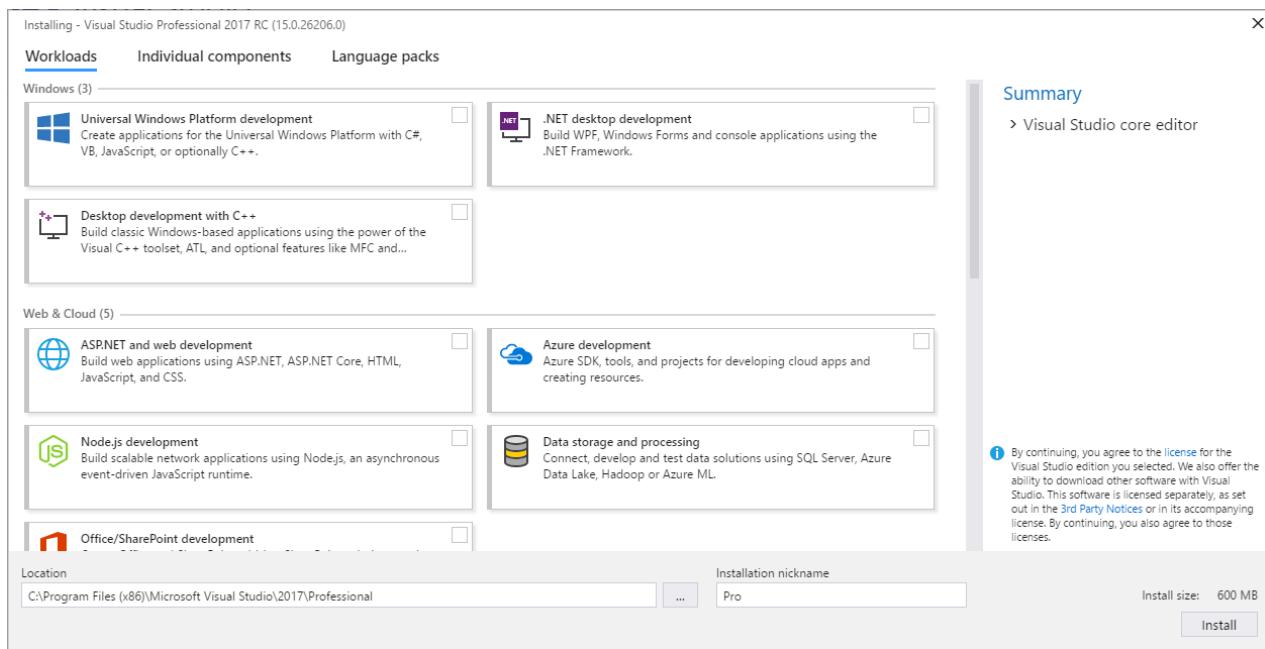


Once the installation is complete, you can [start using Dotfuscator Community](#).

During Visual Studio Installation

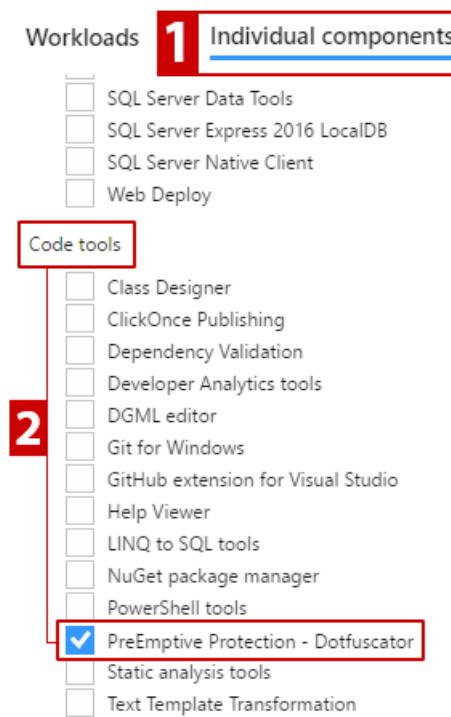
If you have not yet installed Visual Studio, you can obtain the installer from [the Visual Studio website](#). When run, it will display installation options for the selected Visual Studio edition.



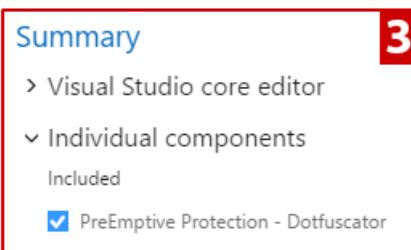


You can then install Dotfuscator Community as an individual component of Visual Studio:

1. Select the **Individual components** tab.
2. Under *Code tools*, check the *PreEmptive Protection - Dotfuscator* item.



3. The *Summary* panel displays *PreEmptive Protection - Dotfuscator* under the *Individual Components* section.



4. Configure any further installation settings as appropriate for your environment.
5. When ready to install Visual Studio, click the *Install* button.

Once the installation is complete, you can start using Dotfuscator Community. For details, see [the Getting Started page of the full Dotfuscator Community User Guide](#).

See also

[This topic in the full Dotfuscator Community User Guide](#)

Upgrade Dotfuscator Community

10/21/2019 • 2 minutes to read • [Edit Online](#)

Dotfuscator Community offers many application protection and hardening features immediately to all developers using Microsoft Visual Studio. However, there are more features available to users who upgrade their version of Dotfuscator.

Registering Dotfuscator Community

Registered users of Dotfuscator Community get access to additional features, such as [command-line support](#), which makes it easy to integrate Dotfuscator Community into your automated build process. Registering also grants access to a built-in tool used for [decoding obfuscated stack traces](#).

Registration is quick, simple, and free of charge. To register Dotfuscator Community, see [the instructions in the full Dotfuscator Community User Guide](#).

Dotfuscator Professional

While Dotfuscator Community provides a basic level of protection, **PreEmptive Protection - Dotfuscator Professional** includes enhanced obfuscation transforms and protection capabilities such as:

- *Intellectual Property Protection*
 - Additional renaming options, including Enhanced Overload Induction™ and randomized identifier selection.
 - Access to enterprise-level obfuscation transforms, including [transforms targeted at defeating automated code decompilation](#).
 - The ability to [obscure sensitive strings](#), making a simple search of the decompiled code impossible.
 - The ability to [discreetly embed ownership and distribution strings into your assemblies](#), allowing you to determine the source of unauthorized software leaks.
 - The ability to [combine multiple assemblies into one](#), making it even more difficult for attackers to determine the roles of code elements, as separation of concerns has been eliminated.
 - The ability to [automatically remove unused code from your application](#), reducing the amount of sensitive code that is shipped.
- *Application Integrity Protection*
 - Additional [application defense behaviors](#).
 - The ability to provide a warning period before an application's end-of-life deadline.
 - The ability to notify application code during an end-of-life warning period or after the deadline.

Dotfuscator Professional is the industry standard [.NET Obfuscator](#) and is suitable for enterprise developers requiring ongoing support, maintenance, and product updates. Additionally, Dotfuscator Professional offers tighter integration with Visual Studio and is licensed for commercial use.

For more information about the advanced application protection features of Dotfuscator Professional, visit PreEmptive Solutions' [Dotfuscator Overview page](#) and [compare it to Dotfuscator Community](#). Fully supported trials are available at [preemptive.com](#).

See also

[This article in the full Dotfuscator Community User Guide](#)

What's new in Visual Studio 2017

10/31/2019 • 20 minutes to read • [Edit Online](#)

Updated for the 15.9 release

Looking to upgrade from a previous version of Visual Studio? Here's what Visual Studio 2017 can offer you: Unparalleled productivity for any dev, any app, and any platform. Use Visual Studio 2017 to develop apps for Android, iOS, Windows, Linux, web, and cloud. Code fast, debug and diagnose with ease, test often, and release with confidence. You can also extend and customize Visual Studio by building your own extensions. Use version control, be agile, and collaborate efficiently with this release!

[DOWNLOAD VISUAL
STUDIO](#)

Here's a high-level recap of the changes that were made since the previous version, Visual Studio 2015:

- **Redefined fundamentals.** A new setup experience means that you can install more quickly and install what you want when you need it.
- **Performance and productivity.** We have focused on new and modern mobile, cloud, and desktop development capabilities. And, Visual Studio starts faster, is more responsive, and uses less memory than before.
- **Cloud app development with Azure.** A built-in suite of Azure tools enable you to easily create cloud-first apps powered by Microsoft Azure. Visual Studio makes it easy to configure, build, debug, package, and deploy apps and services on Azure.
- **Windows app development.** Use the UWP templates in Visual Studio 2017 to create a single project for all Windows 10 devices – PC, tablet, phone, Xbox, HoloLens, Surface Hub, and more.
- **Mobile app development.** Innovate and get results fast with Xamarin, which unifies your multi-platform mobile requirements to one core codebase and set of skills.
- **Cross-platform development.** Seamlessly deliver software to any targeted platform. Extend DevOps processes to SQL Server through Redgate Data Tools and safely automate database deployments from Visual Studio. Or, use .NET Core to write apps and libraries that run unmodified across Windows, Linux, and macOS operating systems.
- **Games development.** With Visual Studio Tools for Unity (VSTU), you can use Visual Studio to write game and editor scripts in C# and then use its powerful debugger to find and fix errors.
- **AI development.** With Visual Studio Tools for AI, you can use the productivity features of Visual Studio to accelerate AI innovation. Build, test, and deploy Deep Learning / AI solutions that seamlessly integrate with Azure Machine Learning for robust experimentation capabilities.

NOTE

For a complete list of new features and functionality in Visual Studio 2017, see the [Current release notes](#). And for a peek at future feature offerings, see the [Preview release notes](#).

Here's more detailed information about some of the most notable improvements and new features in Visual Studio 2017.

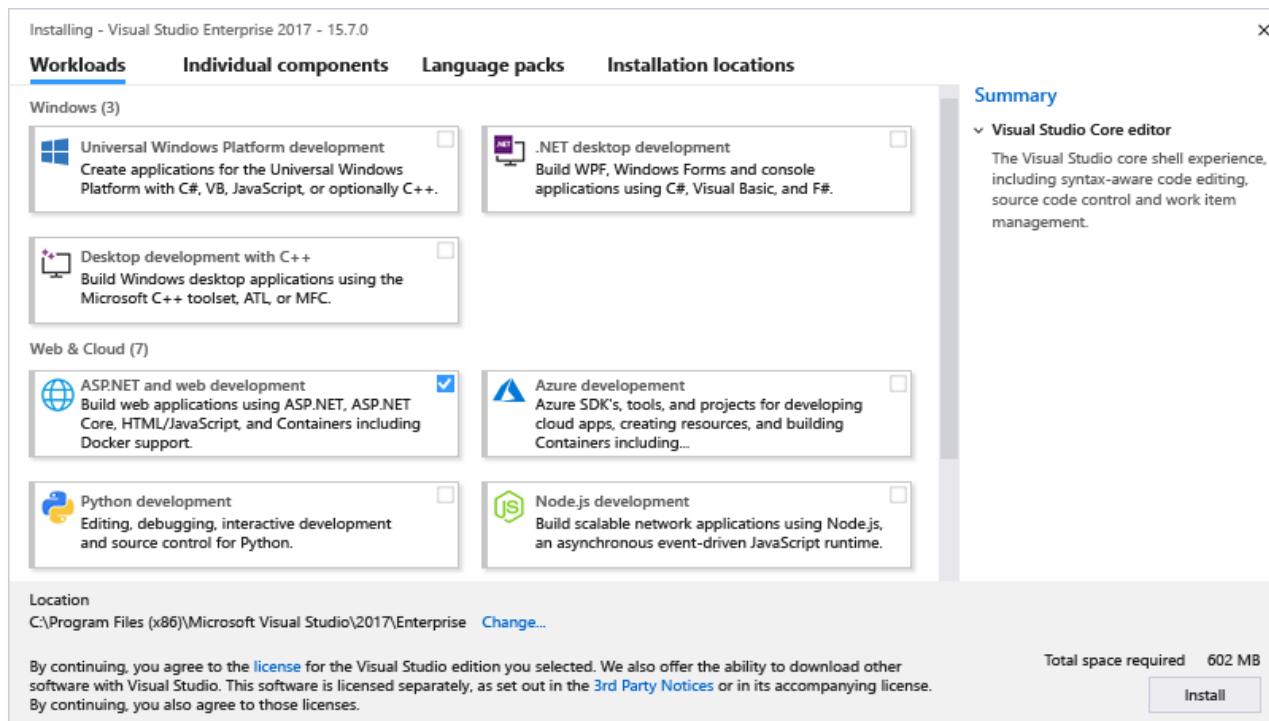
Redefined fundamentals

A new setup experience

Visual Studio makes it easier and faster to install just the features you need, when you need them. And, it uninstalls cleanly, too.

The most important change to note when you install Visual Studio is its new setup experience. On the **Workloads** tab, you'll see installation options that are grouped to represent common frameworks, languages, and platforms. It covers everything from .NET desktop development to C++ application development on Windows, Linux, and iOS.

Choose the workloads you need, and change them when you need to.



And you've got options to fine-tune your installation, too:

- Want to pick your own components instead of using workloads? Select the **Individual components** tab from the installer.
- Want to install Language Packs without also having to change the Windows language option? Choose the **Language packs** tab of the installer.
- New in 15.7:** Want to change the location of where Visual Studio installs? Choose the **Installation options** tab of the installer.

To learn more about the new installation experience, including step-by-step instructions that walk you through it, see the [Install Visual Studio](#) page.

A focus on accessibility

New in 15.3, we made over 1,700 targeted fixes to improve compatibility between Visual Studio and the assistive technologies that many customers use. There are dozens of scenarios that are more compatible with screen readers, high contrast themes, and other assistive technologies than ever before. The debugger, editor, and shell have all gotten significant improvements, too.

For more information, see the [Accessibility improvements in Visual Studio 2017 version 15.3](#) blog post.

Performance and productivity

Sign in across multiple accounts

We've introduced a new identity service in Visual Studio that allows you to share user accounts across Team Explorer, Azure Tools, Microsoft Store publishing, and more.

You can stay signed in longer, too. Visual Studio won't ask you to sign in again every 12 hours. To learn more, see

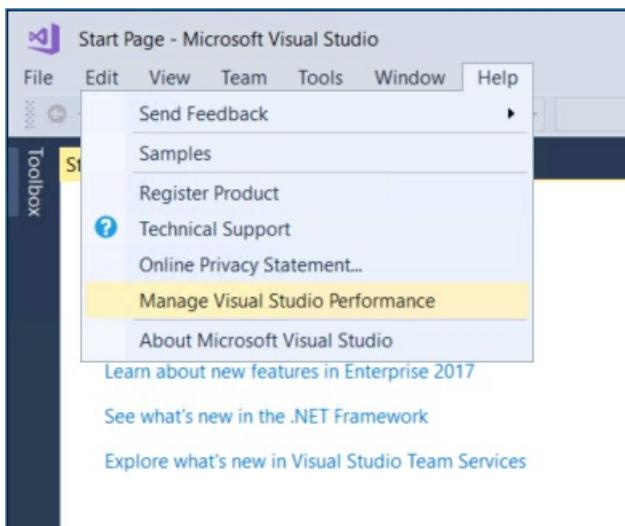
the [Fewer Visual Studio sign-in prompts](#) blog post.

Start Visual Studio faster

The new Visual Studio Performance Center can help you optimize your IDE start-up time. The Performance Center lists all the extensions and tool windows that might slow down the IDE startup. You can use it to improve startup performance by determining when extensions start, or whether tool windows are open at startup.

Faster on-demand loading of extensions

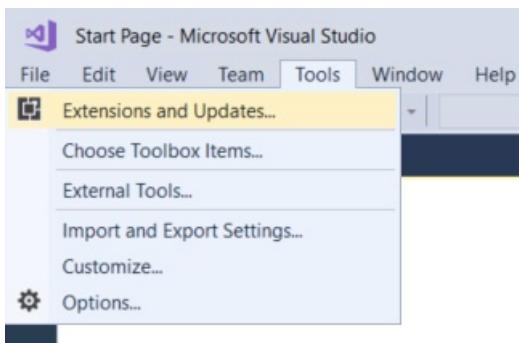
Visual Studio is moving its extensions (and working with third-party extensions too) so that they load on-demand, rather than at IDE startup. Curious about which extensions impact startup, solution load, and typing performance? You can see this information in **Help > Manage Visual Studio Performance**.



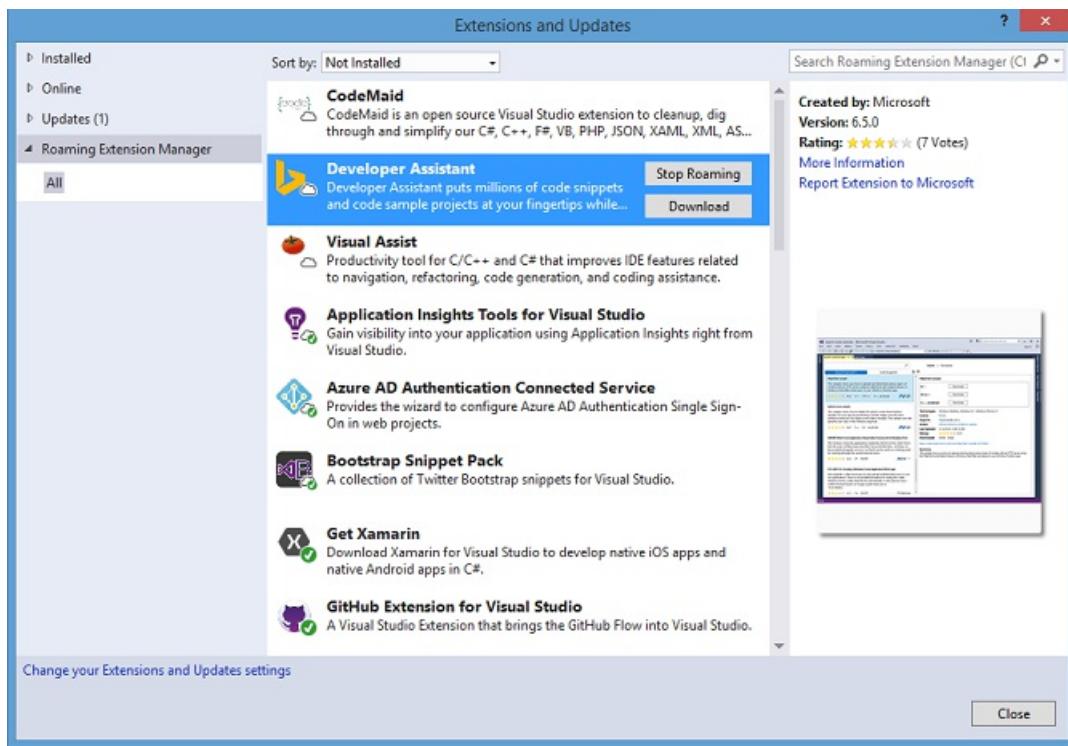
Manage your extensions with Roaming Extensions Manager

It's easier to set up each development environment with your favorite extensions when you sign in to Visual Studio. The new Roaming Extension Manager keeps track of all your favorite extensions by creating a synchronized list in the cloud.

To see a list of your extensions in Visual Studio, click **Tools > Extensions & Updates**, and then click the **Roaming Extension Manager**.



The Roaming Extension Manager tracks all the extensions you install, but you can choose which ones you want to add to your Roaming list.



When you use the Roaming Extension Manager, there are three icon types on your list:

- **Roamed**: An extension that is part of this Roaming List, but not installed on your machine. (You can install these by using the **Download** button.)
- **Roamed & Installed**: All extensions that are part of this Roaming List and installed in your dev environment. (If you decide you do not want to roam, you can remove these by using the **Stop Roaming** button.)
- **Installed**: All extensions that are installed in this environment, but are not part of your Roaming List. (You can add extensions to the Roaming List by using the **Start Roaming** button.)

Any extension that you download while you are signed in is added to your list as **Roamed & Installed**. The extension then becomes part of your Roaming list, which gives you access to it from any machine.

Experience live unit testing

In Visual Studio Enterprise 2017, live unit testing gives you live unit test results and code coverage in the editor while you are coding. It works with C# and Visual Basic projects for both the .NET Framework and .NET Core, and it supports three test frameworks of MSTest, xUnit, and NUnit.

```
1  using System;
2  using System.Text.RegularExpressions;
3
4  namespace UtilityLibraries
5  {
6      public static class StringLibrary
7      {
8          public static bool StartsWithUpper(this String str)
9          {
10             if (String.IsNullOrWhiteSpace(str))
11                 return false;
12
13             Char ch = str[0];
14             return Char.IsUpper(ch);
15         }
16
17         public static bool StartsWithLower(this String str)
18         {
19             if (String.IsNullOrWhiteSpace(str))
20                 return false;
21
22             Char ch = str[0];
23             return Char.IsLower(ch);
24         }
25
26         public static int GetWordCount(this String str)
27         {
28             return Regex.Matches(str, @"\w+").Count;
29         }
30     }
31 }
```

For more information, see the [Introducing Live Unit Testing](#). For a list of new features added in each release of Visual Studio Enterprise 2017, see [What's new in Live Unit Testing](#).

Set up a CI/CD pipeline

Automated testing

Automated testing is a key part of any DevOps pipeline. It allows you to consistently and reliably test and release your solution on much shorter cycles. CI/CD (Continuous Integration and Continuous Delivery) flows can help make the process more efficient.

For more information about automated tests, see the [CI/CD pipeline for automated tests in DevOps](#) blog post.

And, for more information about what's new in the [Continuous delivery tools for Visual Studio](#) DevLabs extension, see the [Commit with confidence: Commit time code quality](#) blog post.

Visual Studio IDE enhancements

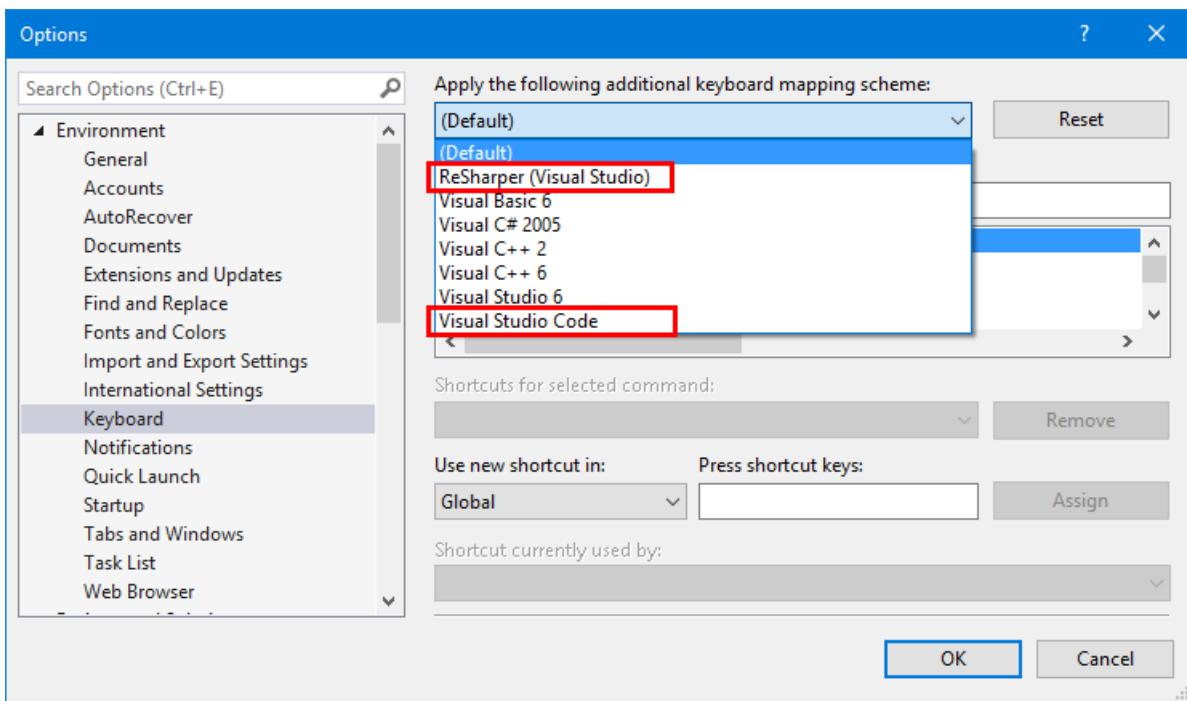
Multi-caret editing

New in 15.8: Editing multiple locations in a file, simultaneously, is now easy. Start by creating insertion points and selections at multiple locations in a file. Then, use the multi-caret editing feature to make the same edit in two or more places at the same time.

For more information, see the [Multi-caret selection](#) section of the of the [Find and replace text](#) page.

Keep keybinding profiles consistent

New in 15.8: Now, you can keep your keybindings consistent across tools with two new keyboard profiles: Visual Studio Code and ReSharper (Visual Studio). You can find these schemes under **Tools > Options > General > Keyboard** and the top drop-down menu.



Use new refactorings

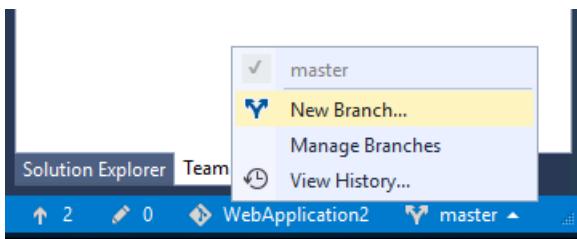
Refactoring is the process of improving your code after it has been written. Refactoring changes the internal structure of the code without changing its behavior. We add new refactorings often; here are just a few:

- Add parameter (from CallSite)
- Generate overrides
- Add named argument
- Add null-check for parameters
- Insert digit-separators into literals
- Change base for numeric literals (for example, hex to binary)
- Convert if-to-switch
- Remove unused variable

For more information, see [Quick Actions](#).

Interact with Git

When you are working with a project in Visual Studio, you can set up and quickly commit and publish your code to a Git service. You can also manage your Git repositories by using menu clicks from buttons in the bottom right-hand corner of the IDE.

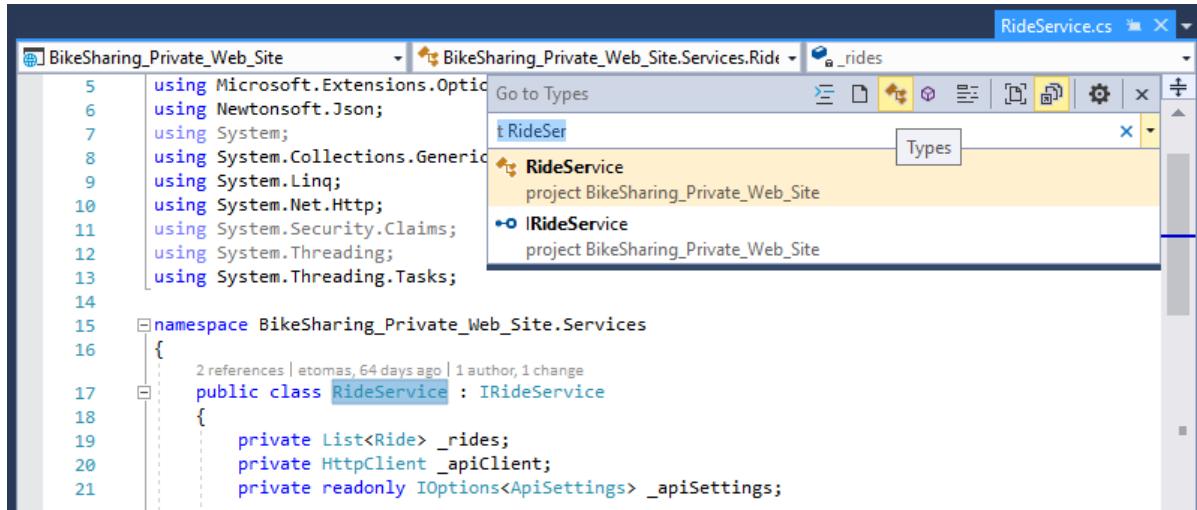


Experience improved navigation controls

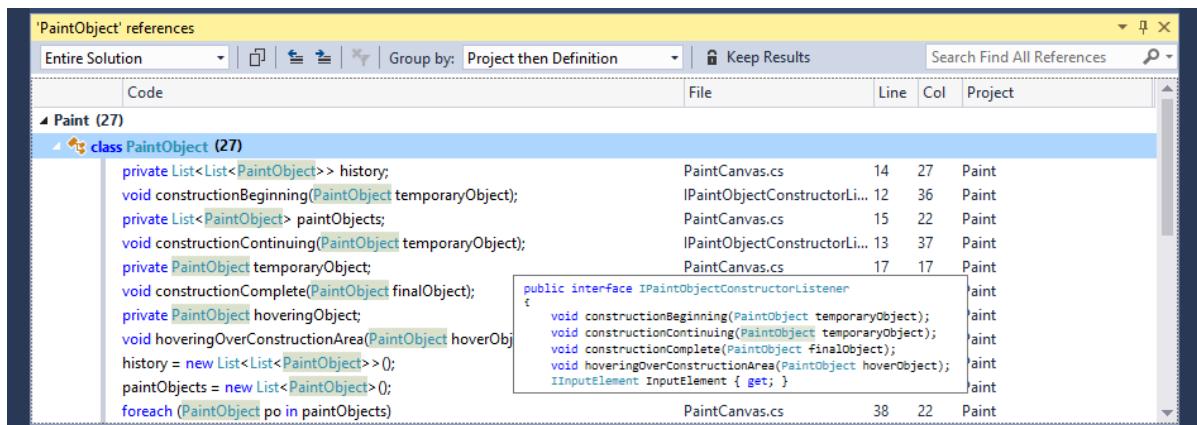
We've refreshed the navigation experience to help you get from A to B with greater confidence and fewer distractions.

- **New in 15.4: Go To Definition (Ctrl+click or F12)** – Mouse users have an easier way to navigate to the definition of a member by pressing **Ctrl** and then clicking the member. Pressing **Ctrl** and hovering over a code symbol will underline it and turn it into a link. See [Go To Definition](#) and [Peek Definition](#) for more information.

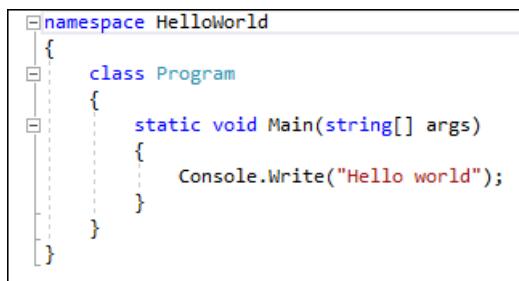
- **Go To Implementation (Ctrl+F12)** – Navigate from any base type or member to its various implementations.
 - **Go To All (Ctrl+T or Ctrl+,)** – Navigate directly to any file/type/member/symbol declaration. You can filter your result list or use the query syntax (for example, "f searchTerm" for files, "t searchTerm" for types, etc.).



- **Find All References (Shift+F12)** – With syntax colorization, you can group Find All Reference results by a combination of project, definition, and path. You can also "lock" results so that you can continue to find other references without losing your original results.



- **Structure Visualizer** – Dotted, gray vertical lines (indent guides) act as landmarks in code to provide context within your frame of view. You may recognize them from the popular Productivity Power Tools. You can use them to visualize and discover what block of code you're in at any time without having to scroll. Hovering over the lines displays a tooltip that shows you the opening of that block and its parents. It's available for all the languages supported via TextMate grammars, as well as C#, Visual Basic, and XAML.



For more information about the new productivity features, see the [Visual Studio 2017: Productivity, Performance, and Partners](#) blog post.

Visual C++

You'll see several improvements in Visual Studio, such as distributing C++ Core Guidelines with Visual Studio,

updating the compiler by adding enhanced support for C++11 and C++ features, and adding and updating functionality in the C++ libraries. We've also improved the performance of the C++ IDE, installation workloads, and more.

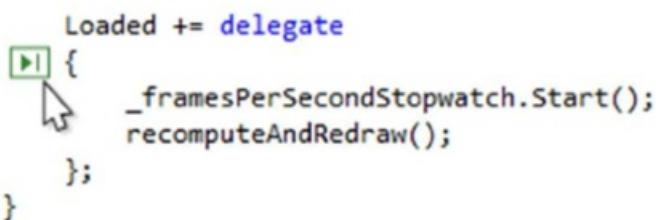
As well, we've fixed over 250 bugs and reported issues in the compiler and tools, many submitted by customers through the [Developer Community for C++](#).

For complete details, see the [What's new for Visual C++ in Visual 2017](#) page.

Debugging and diagnostics

Run to Click

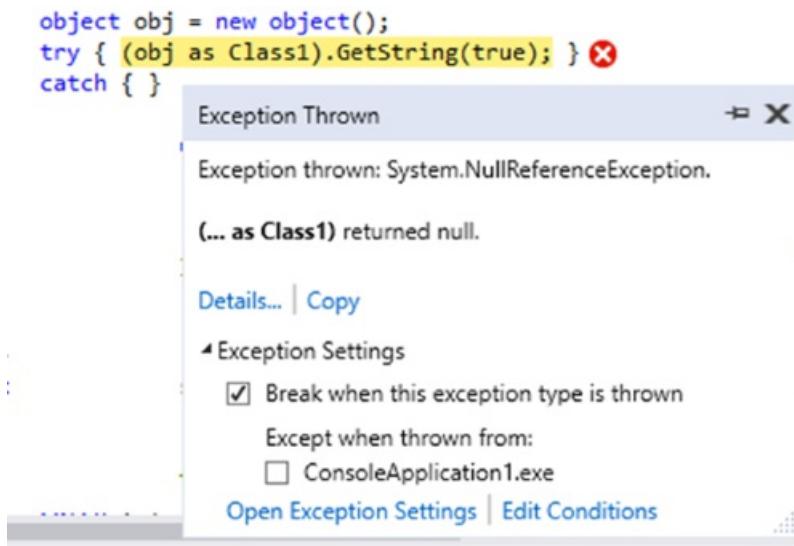
Now, you can more easily skip ahead during debugging without setting a breakpoint to stop on the line you want. When you are stopped in the debugger, just click the icon that appears next to the line of code. Your code will run and stop on that line the next time it is hit in your code path.



```
Loaded += delegate
{
    _framesPerSecondStopwatch.Start();
    recomputeAndRedraw();
};
```

The new Exception Helper

The new Exception Helper helps you view your exception information at-a-glance. The information is presented in a compact form with instant access to inner exceptions. When you diagnose a NullReferenceException, you can quickly see what was null right inside the Exception Helper.



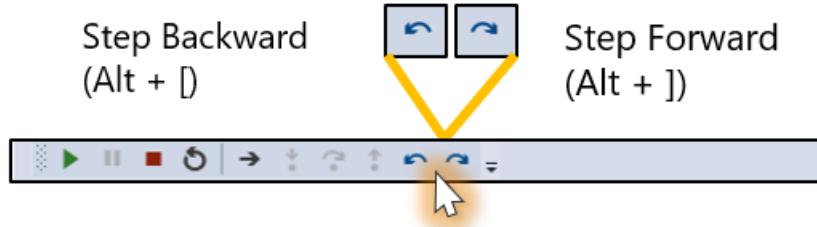
For more information, see the [Use the new Exception Helper in Visual Studio](#) blog post.

Snapshots and IntelliTrace step-back

New in 15.5: IntelliTrace step-back automatically takes a snapshot of your application at every breakpoint and debugger step event. The recorded snapshots enable you to go back to previous breakpoints or steps and view the state of the application as it was in the past. IntelliTrace step-back can save you time when you want to see the previous application state but don't want to restart debugging or recreate the desired app state.

You can navigate and view snapshots by using the **Step Backward** and **Step Forward** buttons in the **Debug** toolbar. These buttons navigate the events that appear in the **Events** tab in the **Diagnostic Tools** window.

Stepping backward or forward to an event automatically activates historical debugging on the selected event.



For more information, see the [View snapshots using IntelliTrace step-back](#) page.

Containerization

Containers provide you with increased app density and lower deployment cost along with improved productivity and DevOps agility.

Docker Container Tooling

New in 15.5:

- Visual Studio includes tools for Docker containers that now support multi-stage Dockerfiles, which streamline creating optimized container images.
- By default, Visual Studio will automatically pull, build, and run the necessary container images in the background when you open a project that has Docker support. You can disable this via the **Automatically start containers in background** setting in Visual Studio.

Cloud app development with Azure

Azure Functions tools

As part of the "Azure development" workload, we've included tools to help you develop Azure functions by using pre-compiled C# class libraries. Now you can build, run, and debug on your local development machine and then publish directly to Azure from Visual Studio.

For more information, see the [Azure Functions tools for Visual Studio](#) page.

Debug live ASP.NET apps using snappoints and logpoints in live Azure applications

New in 15.5: The Snapshot Debugger takes a snapshot of your in-production apps when code that you are interested in executes. To instruct the debugger to take a snapshot, you set snappoints and logpoints in your code. The debugger lets you see exactly what went wrong, without impacting traffic of your production application. The Snapshot Debugger can help you dramatically reduce the time it takes to resolve issues that occur in production environments.

Snapshot collection is available for the following web apps running in Azure App Service:

- ASP.NET applications running on .NET Framework 4.6.1 or later.
- ASP.NET Core applications running on .NET Core 2.0 or later on Windows.

For more information, see [Debug live ASP.NET apps using snappoints and logpoints](#).

Windows app development

Universal Windows Platform

The Universal Windows Platform (UWP) is the app platform for Windows 10. You can develop apps for UWP with just one API set, one app package, and one store to reach all Windows 10 devices – PC, tablet, phone, Xbox, HoloLens, Surface Hub, and more. UWP supports different screen sizes and a variety of interaction models, whether it be touch, mouse and keyboard, a game controller, or a pen. At the core of UWP apps is the idea that

users want their experiences to be mobile across ALL their devices, and that they want to use whatever device is most convenient or productive for the task at hand.



Choose your preferred development language—from C#, Visual Basic, C++, or JavaScript—to create a Universal Windows Platform app for Windows 10 devices. Visual Studio 2017 provides a UWP app template for each language that lets you create a single project for all devices. When your work is finished, you can produce an app package and submit it to Microsoft Store from within Visual Studio to get your app out to customers on any Windows 10 device.

New in 15.5: Visual Studio 2017 version 15.5 provides the best support for the Windows 10 Fall Creators Update SDK (10.0.16299.0). The Windows 10 Fall Creators Update also brings many improvements for UWP developers. Here are some of the biggest changes:

- **Support for .NET Standard 2.0**

In addition to streamlined app deployment, the Windows 10 Fall Creators Update is the first release of Windows 10 to provide .NET Standard 2.0 support. Effectively, [.NET Standard](#) is a reference implementation of the base class library that any .NET platform can implement. The goal of .NET Standard is to make it as easy as possible for .NET developers to share code across any .NET platform they choose to work on.

- **The best of both UWP and Win32**

We have improved the Windows 10 Platform with the [Desktop Bridge](#) to make Windows 10 better for all .NET developers, whether their current focus is on UWP, WPF, Windows Forms, or Xamarin. With the new App Packaging project type in Visual Studio 2017 version 15.5, you can create Windows App Packages for your WPF or Windows Forms projects, just like you can for UWP projects. After you package your app, you get all the Windows 10 app deployment benefits and have the option to distribute via Microsoft Store (for consumer apps) or Microsoft Store for Business and Education. Because packaged apps have access to both the full UWP API surface and the Win32 APIs on desktop, you can now modernize your WPF and Windows Forms applications gradually with UWP APIs and Windows 10 features. Moreover, you can include your Win32 components in your UWP applications that light up on desktop with all Win32 capabilities.

For more information about UWP, see the [Develop apps for the Universal Windows Platform \(UWP\)](#) page.

Mobile app development

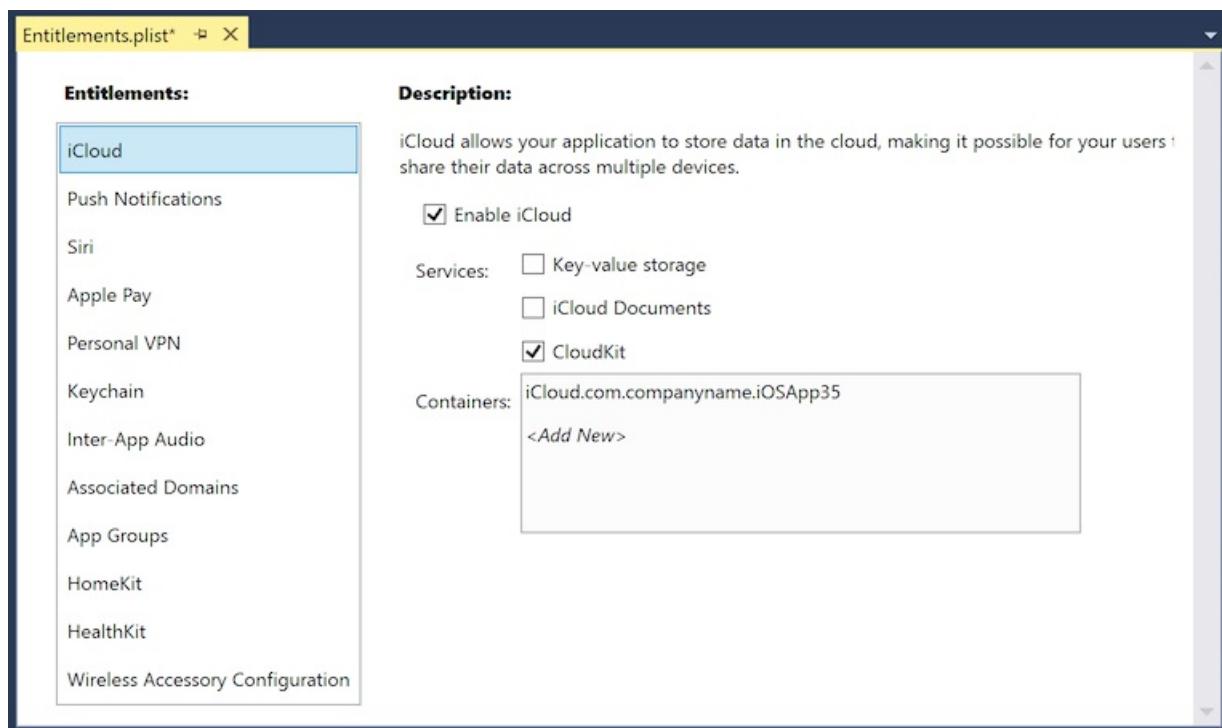
Xamarin

As part of the "Mobile development with .NET" workload, developers familiar with C#, .NET, and Visual Studio can deliver native Android, iOS, and Windows apps by using Xamarin. Developers can enjoy the same power and productivity when working with Xamarin for mobile apps, including remote debugging on Android, iOS, and Windows devices—without having to learn native coding languages like Objective-C or Java.

For more information, see the [Visual Studio and Xamarin](#) page.

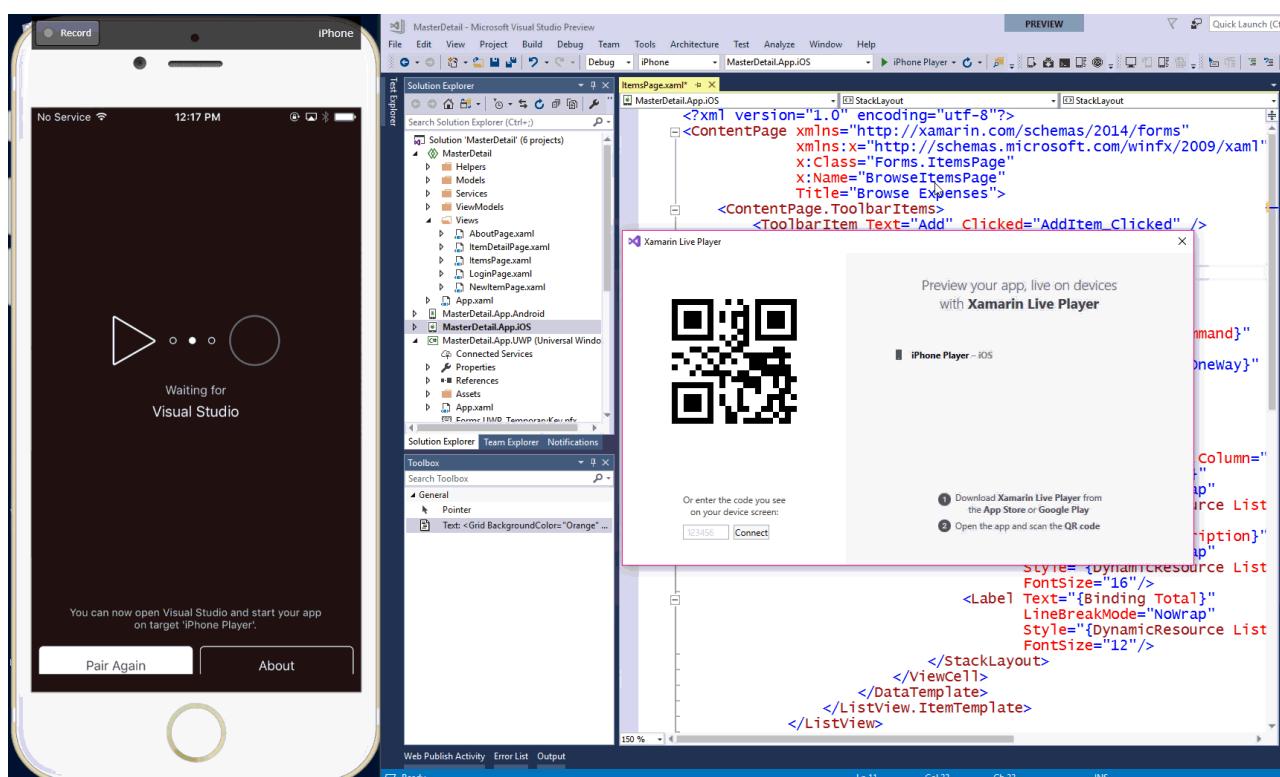
Entitlements editor

New in 15.3: For your iOS development needs, we've added a stand-alone Entitlements editor. It includes a user-friendly UI that can be easily browsed. To launch it, double-click your *entitlements.plist* file.



Visual Studio Tools for Xamarin

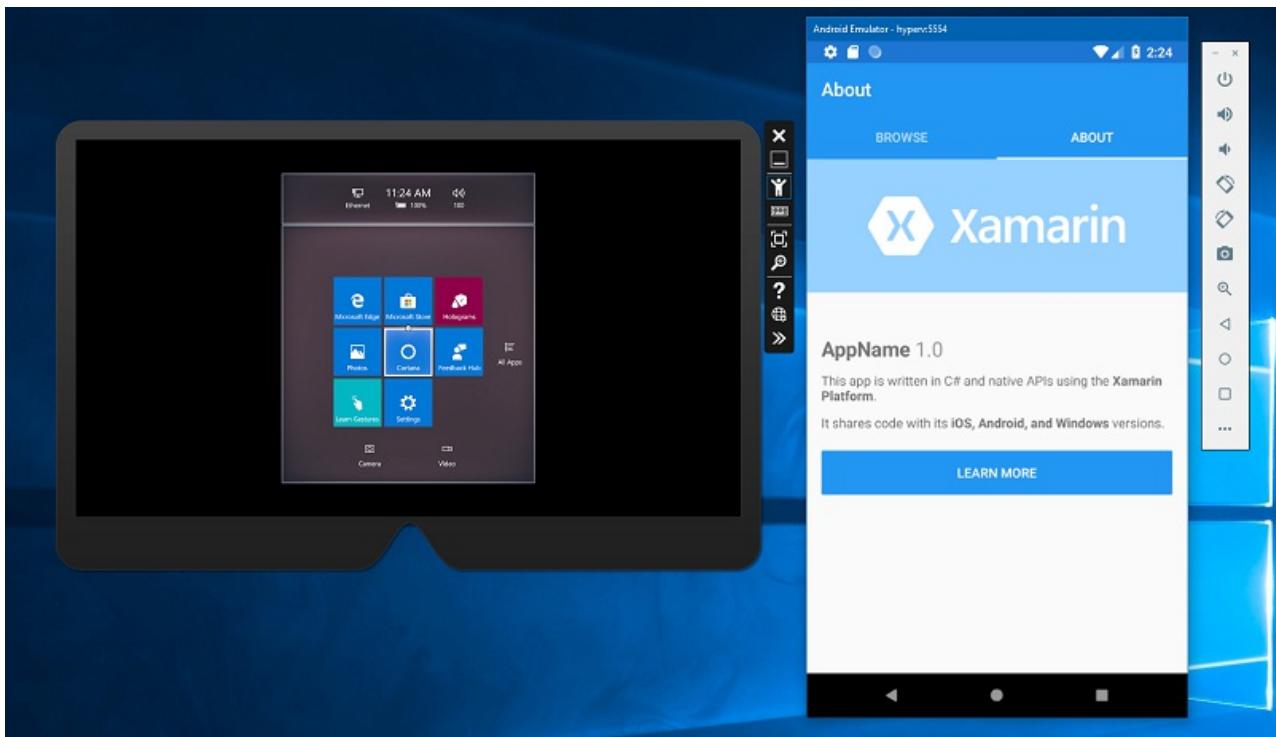
New in 15.4: Xamarin Live enables developers to continuously deploy, test, and debug their apps, directly on iOS and Android devices. After you download the Xamarin Live Player—available in the App Store or on Google Play—you can pair your device with Visual Studio and revolutionize the way you build mobile apps. This functionality is now included in Visual Studio and can be enabled by going to **Tools > Options > Xamarin > Other > Enable Xamarin Live Player**.



Support for Google Android Emulator

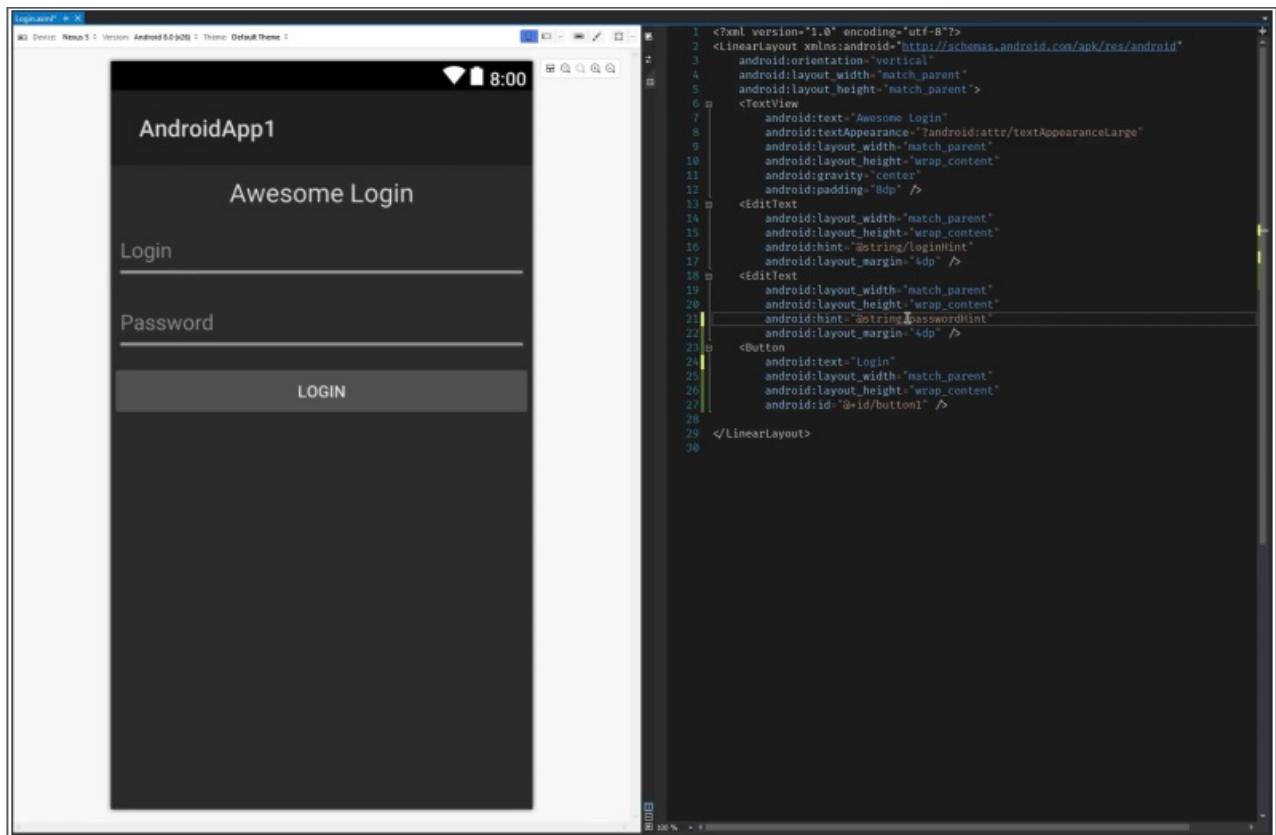
New in 15.8: When you use Hyper-V, now you can use Google's Android Emulator side-by-side with other

technologies that are based on Hyper-V, such as Hyper-V virtual machines, Docker tooling, the HoloLens emulator, and more. (This feature requires Windows 10 April 2018 Update or later.)



Xamarin.Android Designer split-view editor

Also **new in 15.8**: We've made significant improvements to the designer experience for Xamarin.Android. A highlight is the new split-view editor that allows you to create, edit, and preview your layouts at the same time.



For more information, see [Hardware acceleration for emulator performance](#)

Visual Studio App Center

New in 15.5: Visual Studio App Center—which is now generally available for Android, iOS, macOS, and Windows apps—has everything you need to manage the lifecycle of your apps, including automated builds, testing on real devices in the cloud, distribution to beta testers and app stores, and monitoring of real-world usage through crash

and analytics data. Apps written in Objective-C, Swift, Java, C#, Xamarin, and React Native are supported across all features.

The screenshot shows the Microsoft App Center interface. On the left, there's a sidebar with options like ContosoAir-iOS, Getting Started, Build, Test (which is selected), Test runs, Device sets, Distribute, Crashes, Analytics, Push, and Settings. Below that is a user profile for Keith-Ballinger. The main area shows a test run titled "Side Menu Navigation Test" from September 14, 2017, at 5:54:16 PM, with 3 tests. A green checkmark indicates success for the "Side Menu Navigation Test". The test details show steps such as "On LoginPage", "Credentials Entered", "On HomePage", "Selecting Option Func'2", "On MyTripsPage", "Selecting Option Func'2", "On ContactPage", and "Selecting Option Func'2". Below the test details, there are 18 device screenshots arranged in a grid, showing the app running on various iOS devices including iPhone 7, iPhone 6s, iPad Pro, and iPad Mini models with different iOS versions (e.g., 9.3.5, 10.0.2, 10.3.3).

For more information, see the [Introducing App Center: Build, test, distribute and monitor apps in the cloud](#) blog post.

Cross-platform development

Redgate Data Tools

To extend DevOps capabilities to SQL Server database development, Redgate Data Tools are now available in Visual Studio.

Included with Visual Studio 2017 Enterprise:

- [Redgate ReadyRoll Core](#) helps you develop migration scripts, manage database changes using source control, and safely automate deployments of SQL Server database changes alongside applications changes.
- [Redgate SQL Prompt Core](#) helps you write SQL more quickly and accurately with the help of intelligent code completion. SQL Prompt autocompletes database and system objects and keywords, and offers column suggestions as you type. This results in cleaner code and fewer errors because you don't have to remember every column name or alias.

Included with all editions of Visual Studio 2017:

- [Redgate SQL Search](#) increases your productivity by helping you quickly find SQL fragments and objects across multiple databases.

To learn more, see the [Redgate Data Tools in Visual Studio 2017](#) blog post.

.NET Core

.NET Core is a general purpose, modular, cross-platform, and open source implementation of the .NET Standard and contains many of the same APIs as the .NET Framework.

The .NET Core platform is made of several components, which include the managed compilers, the runtime, the base class libraries, and numerous application models, such as ASP.NET Core. .NET Core supports three main

operating systems: Windows, Linux, and macOS. You can use .NET Core in device, cloud, and embedded/IoT scenarios.

And, it now includes Docker support.

New in 15.3: Visual Studio 2017 version 15.3 supports .NET Core 2.0 development. Using .NET Core 2.0 requires downloading and installing the .NET Core 2.0 SDK separately.

For more information, see the [.NET Core guide](#) page.

Games development

Visual Studio Tools for Unity

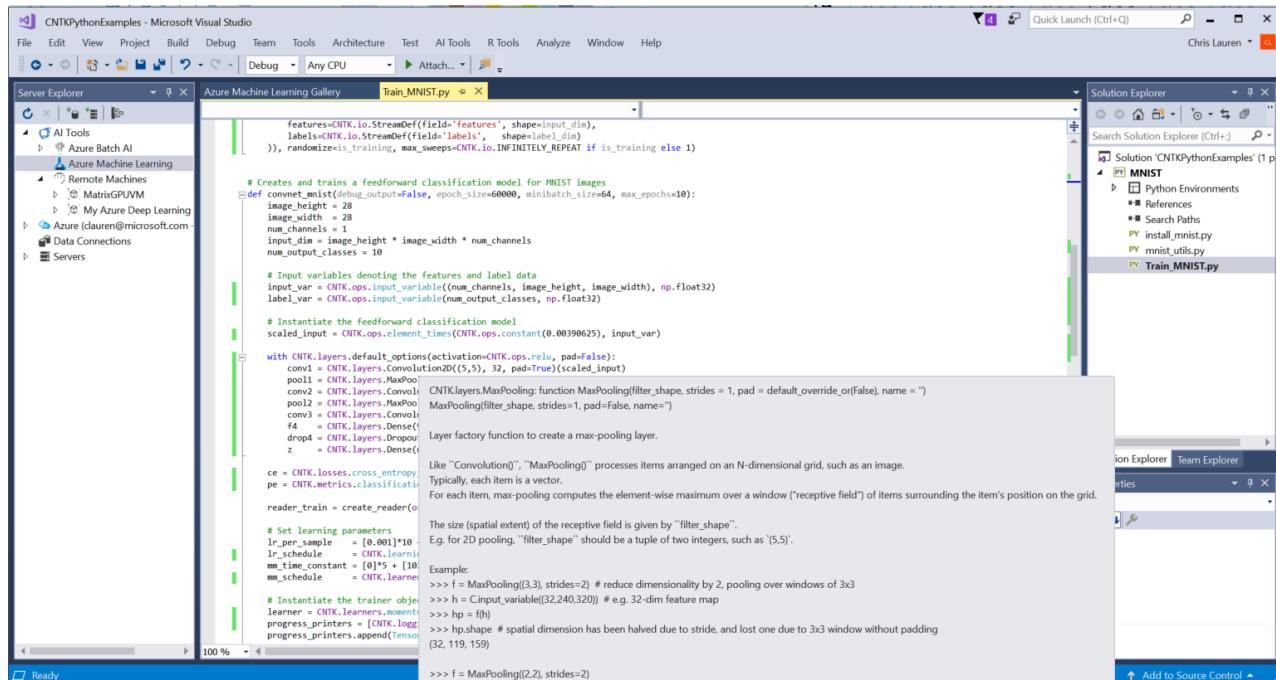
As part of the "Games development for Unity" workload, we've included tools to help you develop cross-platform to create 2D and 3D games and interactive content. Create once and publish to 21 platforms, including all mobile platforms, WebGL, Mac, PC and Linux desktop, web, or consoles by using Visual Studio 2017 and Unity 5.6.

For more information, see the [Visual Studio Tools for Unity](#) page.

AI development

Visual Studio Tools for AI

New in 15.5: Use the productivity features of Visual Studio to accelerate AI innovation today. Use built-in code editor features like syntax highlighting, IntelliSense, and text auto formatting. You can interactively test your deep learning application in your local environment by using step-through debugging on local variables and models.



The screenshot shows the Microsoft Visual Studio 2017 interface with the following details:

- Title Bar:** CNTKPythonExamples - Microsoft Visual Studio
- Menu Bar:** File, Edit, View, Project, Build, Debug, Team, Tools, Architecture, Test, AI Tools, R Tools, Analyze, Window, Help
- Toolbars:** Standard, Status
- Toolbox:** Standard
- Server Explorer:** Shows Azure Machine Learning, Remote Machines, and Servers.
- Azure Machine Learning Gallery:** Train_MNIST.py
- Code Editor:** Displays Python code for training a feedforward classification model on MNIST images using the CNTK library. The code includes imports for CNTK, numpy, and other modules, defines variables for features and labels, creates a data stream, instantiates a feedforward model, sets learning parameters, and trains the model.
- Solution Explorer:** Shows the solution 'CNTKPythonExamples' with a project 'MNIST' containing files like 'install_minist.py', 'mnist_utils.py', and 'Train_MNIST.py'.
- Properties Explorer:** Standard properties view.
- Task List:** Standard task list view.
- Output:** Standard output window.
- Team Explorer:** Standard team explorer view.
- Activity Bar:** Add to Source Control, Ready

For more information, see the [Visual Studio Tools for AI](#) page.

What's next

We update Visual Studio 2017 often with new features that can make your development experience even better. Here's a recap of some of our most notable updates that are in experimental preview:

- **Live Share**, a new tool that allows you to share a codebase and its context with a teammate and get instant bidirectional collaboration directly from within Visual Studio. With Live Share, a teammate can read, navigate, edit, and debug a project that you've shared with them, and do so seamlessly and securely.

For more information, see the [Live Share FAQ](#).

- **IntelliCode**, a new capability that enhances software development by using AI to deliver better context-aware code completions, guide developers to code to the patterns and styles of their team, find difficult-to-catch code issues, and focus code reviews on areas that really matter.

For more information, see the [IntelliCode FAQ](#).

Want to know more about what else is in the works for Visual Studio 2017? See the [Visual Studio Roadmap](#) page.

And don't forget to check out our newest version, [Visual Studio 2019](#).

Contact us

Why send feedback to the Visual Studio team? Because we take customer feedback seriously. It drives much of what we do.

If you want to make a suggestion about how we can improve Visual Studio, or learn more about product support options, please see the [Send us feedback](#) page.

Report a problem

Sometimes, a message isn't enough to convey the full impact of a problem that you've encountered. If you experience a hang, crash, or other performance issue, you can easily share repro steps and supporting files (such as screenshots, and trace and heap dump files) with us by using the **Report a Problem** tool. For more information about how to use this tool, see the [How to report a problem](#) page.

See also

- [Visual Studio 2017 release notes](#)
- [What's New in the Visual Studio 2017 SDK](#)
- [What's new in Visual C++](#)
- [What's new in C#](#)
- [What's new for Team Foundation Server](#)
- [What's new in Visual Studio for Mac](#)
- [What's new in Visual Studio 2019](#)

What's new in Visual Studio 2019

12/4/2019 • 8 minutes to read • [Edit Online](#)

Updated for the 16.4 release

[DOWNLOAD VISUAL STUDIO
2019](#)

With Visual Studio 2019, you'll get best-in-class tools and services for any developer, any app, and any platform. Whether you're using Visual Studio for the first time or you've been using it for years, there's a lot to like in this new version!

Here's a high-level recap of what's new:

- **Develop:** Stay focused and productive with improved performance, instant code cleanup, and better search results.
- **Collaborate:** Enjoy natural collaboration through a Git-first workflow, real-time editing and debugging, and code reviews right in Visual Studio.
- **Debug:** Highlight and navigate to specific values, optimize memory use, and take automatic snapshots of your application's execution.

For a complete list of everything that's new in this version, see the [release notes](#).

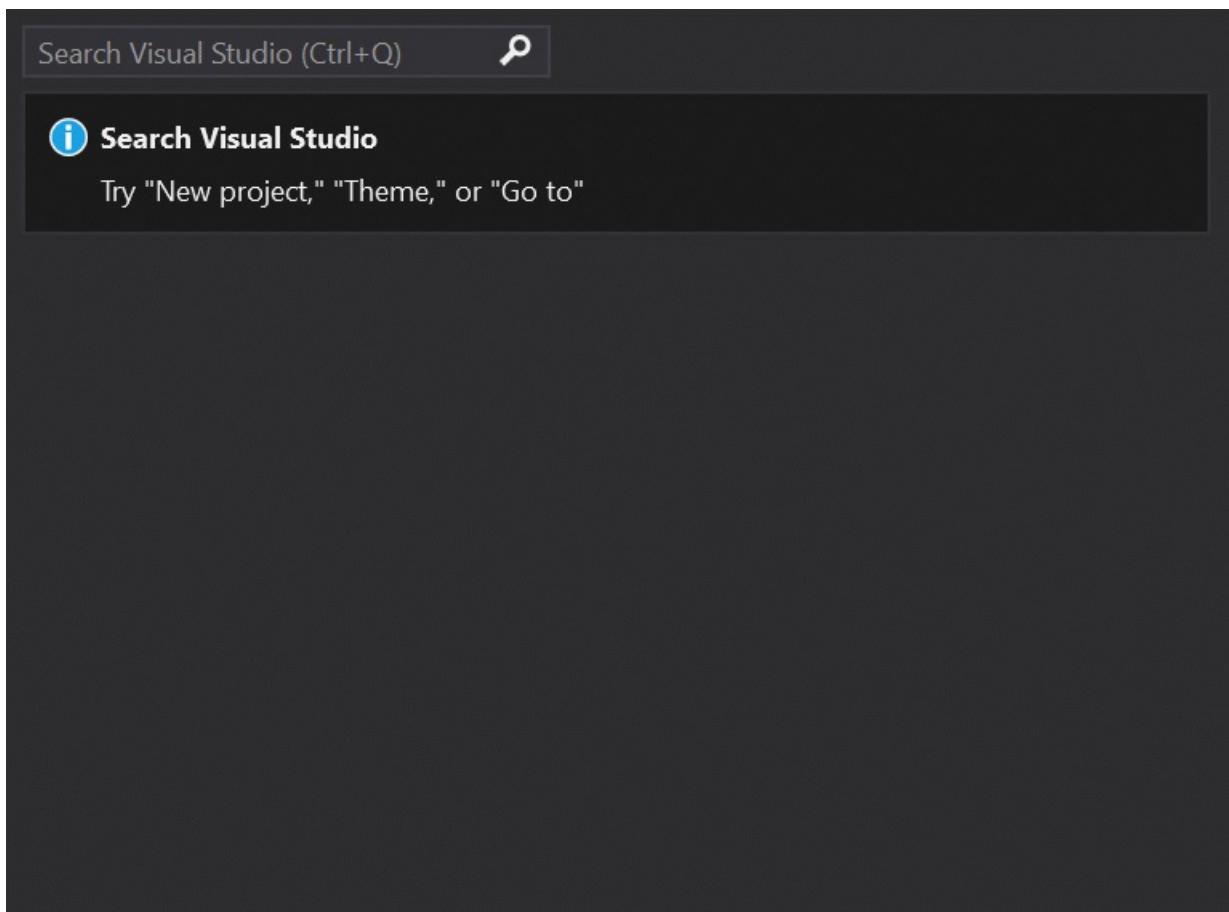
Develop

View the following video to learn more about how you can save time with new features.

Video length: 3.00 minutes

Improved search

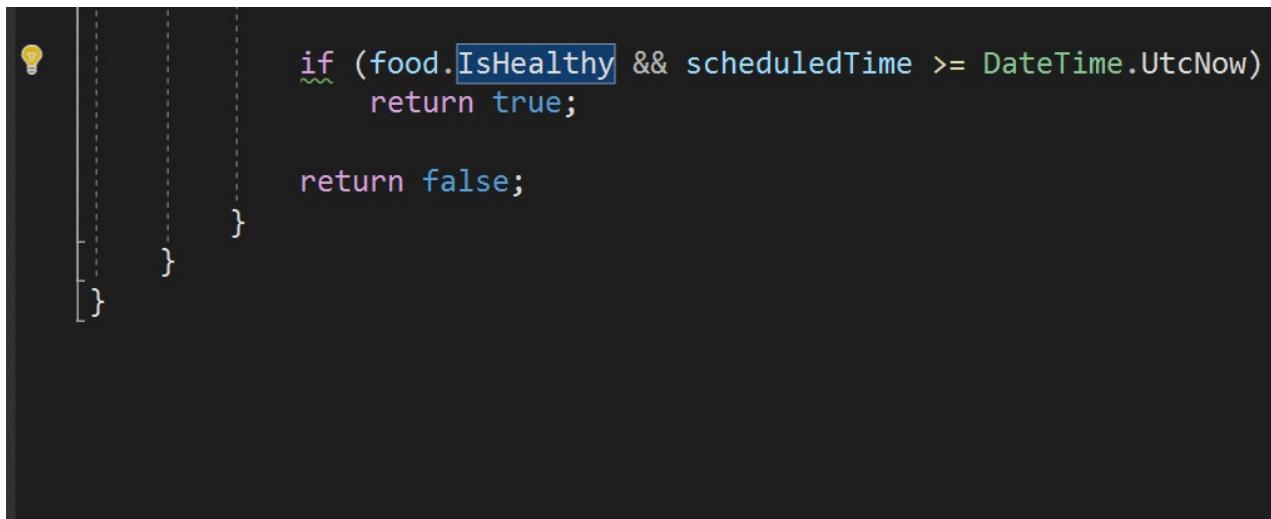
Formerly known as Quick Launch, our new search experience is faster and more effective. Now, search results appear dynamically as you type. And, search results can often include keyboard shortcuts for commands, so that you can more easily memorize them for future use.



The new fuzzy search logic will find anything you need, regardless of typos. So, whether you're looking for commands, settings, documentation, or other useful things, the new search feature makes it easier to find what you're looking for.

Refactorings

There are lots of new and highly useful refactorings in C# that make it easier to organize your code. They show up as suggestions in the light bulb and include actions such as moving members to interface or base class, adjusting namespaces to match folder structure, convert foreach-loops to Linq queries, and more.



Simply invoke the refactorings by pressing **Ctrl+.** and selecting the action you want to take.

IntelliCode

[Visual Studio IntelliCode](#) enhances your software development efforts by using artificial intelligence (AI). IntelliCode trains across 2,000 open-source projects on GitHub—each with over 100 stars—to generate its recommendations.

```
0 references
public void EatBreakfast()
{
    var banana = new Banana();
}

0 references
public bool Eat(IFood food, DateTime scheduledTime)
{
```

Here are a few ways that Visual Studio IntelliCode can help enhance your productivity:

- Deliver context-aware code completions
- Guide developers to adhere to the patterns and styles of their team
- Find difficult-to-catch code issues
- Focus code reviews by drawing attention to areas that really matter

We initially supported only C# when we first previewed the IntelliCode as an extension for Visual Studio. Now, **new in 16.1**, we've added support for C# and XAML "in-the-box". (Support for C++ and TypeScript/JavaScript are still in preview, however.)

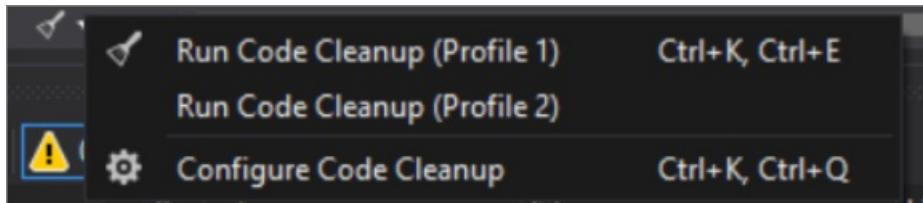
And if you're using C#, we've also added the ability to train a custom model on your own code.

For more information about IntelliCode, see the [Announcing the general availability of IntelliCode plus a sneak peek](#) and [Code more, scroll less with Visual Studio IntelliCode](#) blog posts.

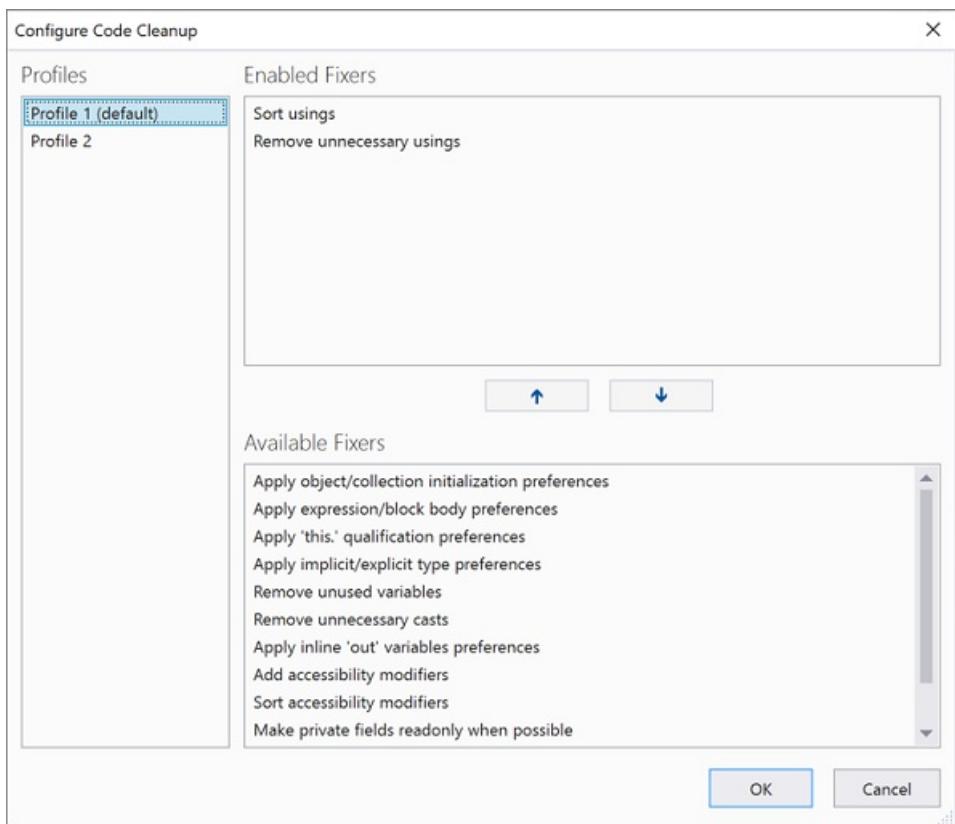
Code cleanup

Paired with a new document health indicator is a new code cleanup command. You can use this new command to identify and then fix both warnings and suggestions with the click of a button.

The cleanup will format the code and apply any code fixes as suggested by the [current settings](#) and [.editorconfig files](#).



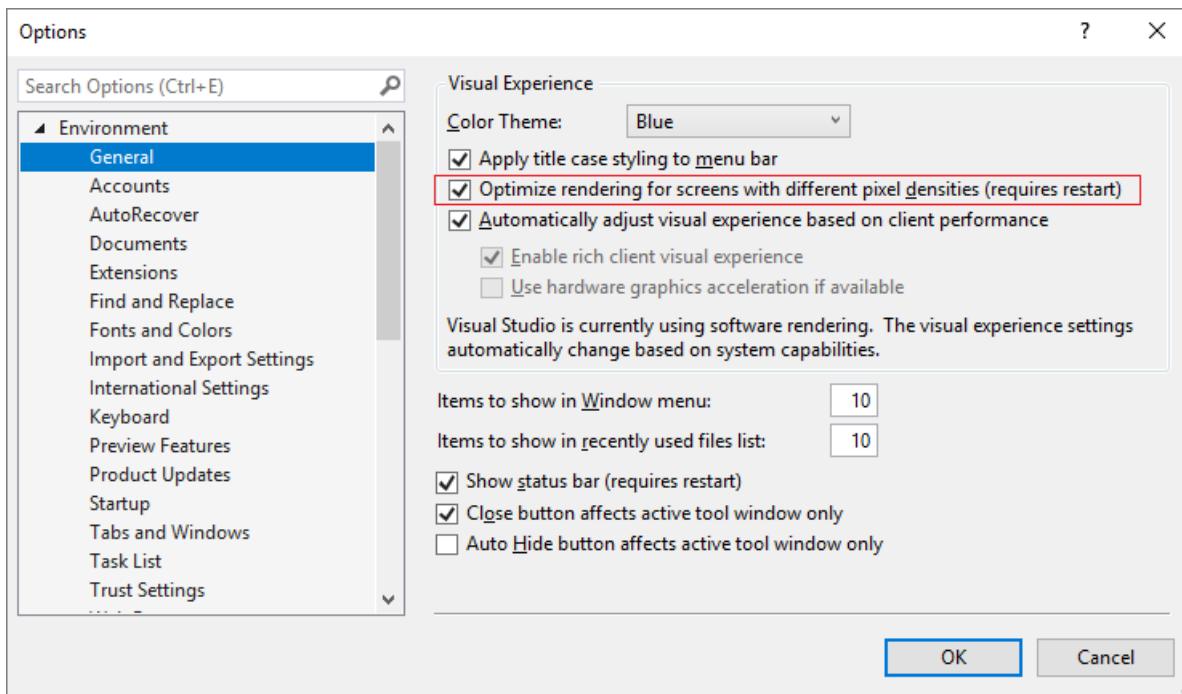
You can also save collections of fixers as a profile. For example, if you have a small set of targeted fixers that you apply frequently while you code, and then you have another comprehensive set of fixers to apply before a code review, you can configure profiles to address these different tasks.



Per-monitor aware (PMA) rendering

If you use monitors that are configured with different display scale factors, or connect remotely to a machine with display scale factors that are different from your main device, you might notice that Visual Studio looks blurry or renders at the wrong scale.

With the release of Visual Studio 2019, we're making Visual Studio a per-monitor aware (PMA) application. Now, Visual Studio renders correctly regardless of the display scale factors you use.

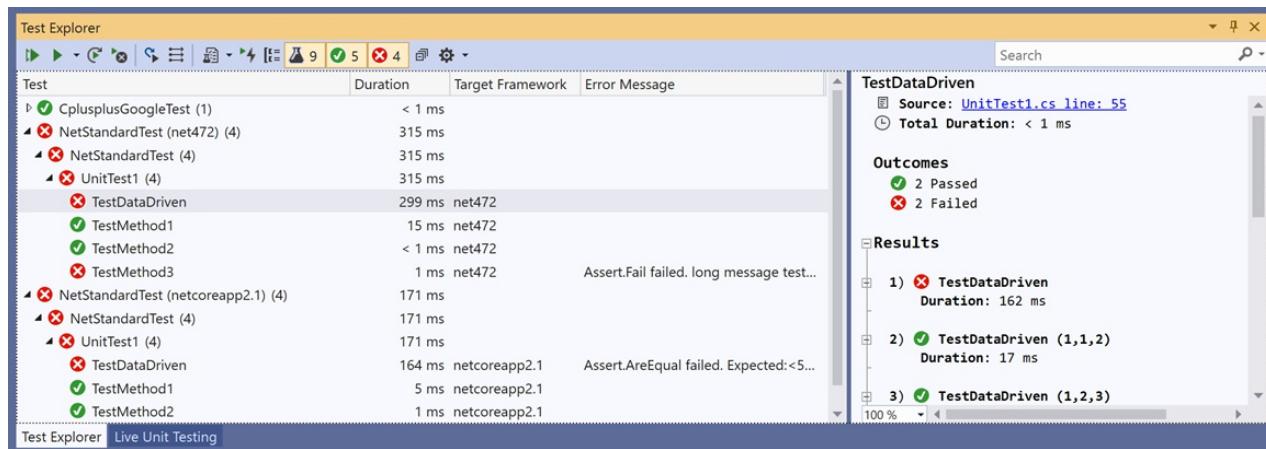


For more information, see the [Better multi-monitor experience with Visual Studio 2019](#) blog post.

Test Explorer

New in 16.2: We've updated Test Explorer to provide better handling of large test sets, easier filtering, more discoverable commands, tabbed playlist views, and customizable columns that let you fine-tune what test

information is displayed.



.NET Core

New in 16.3: We've included support for .NET Core 3.0. Cross-platform, open source—and fully supported by Microsoft.

For more information, see the [Announcing .NET Core 3.0](#) blog post.

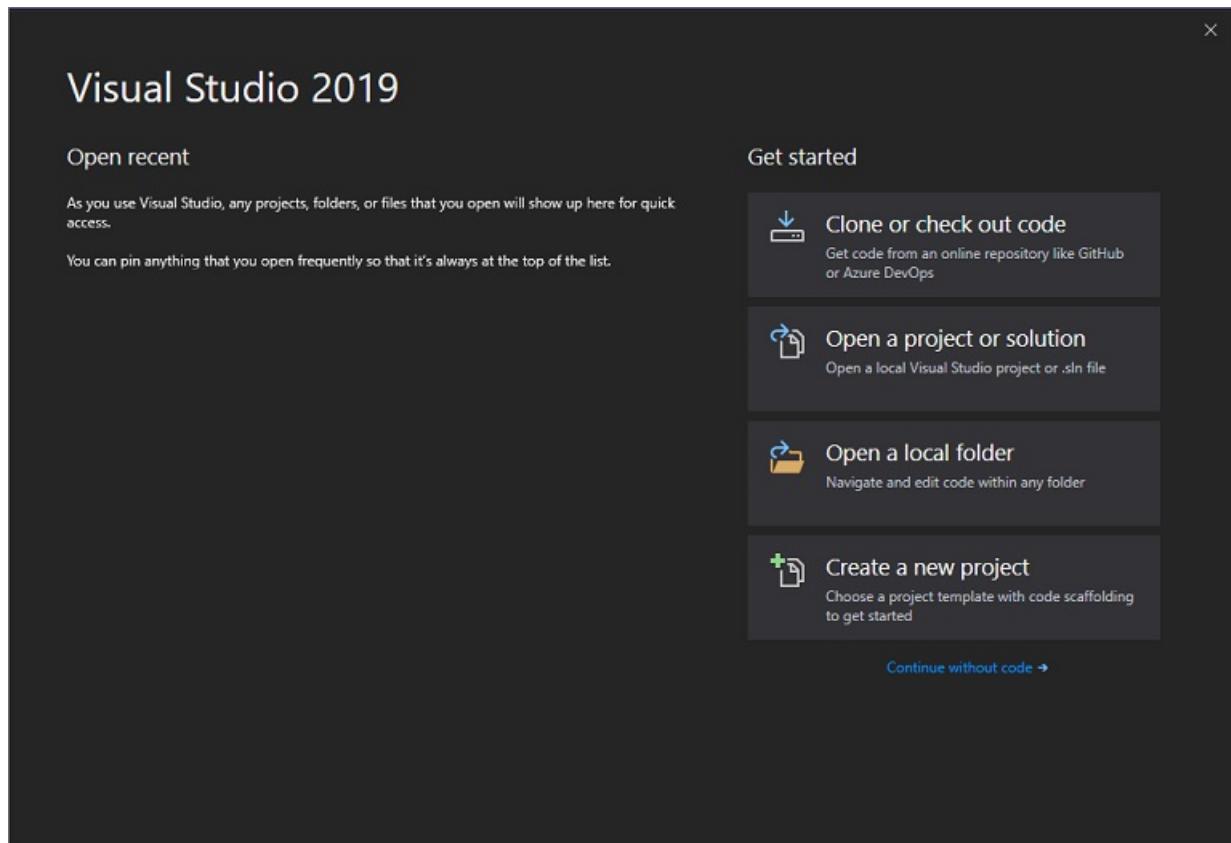
Collaborate

View the following video to learn more about how you can team up to solve issues.

Video length: 4.22 minutes

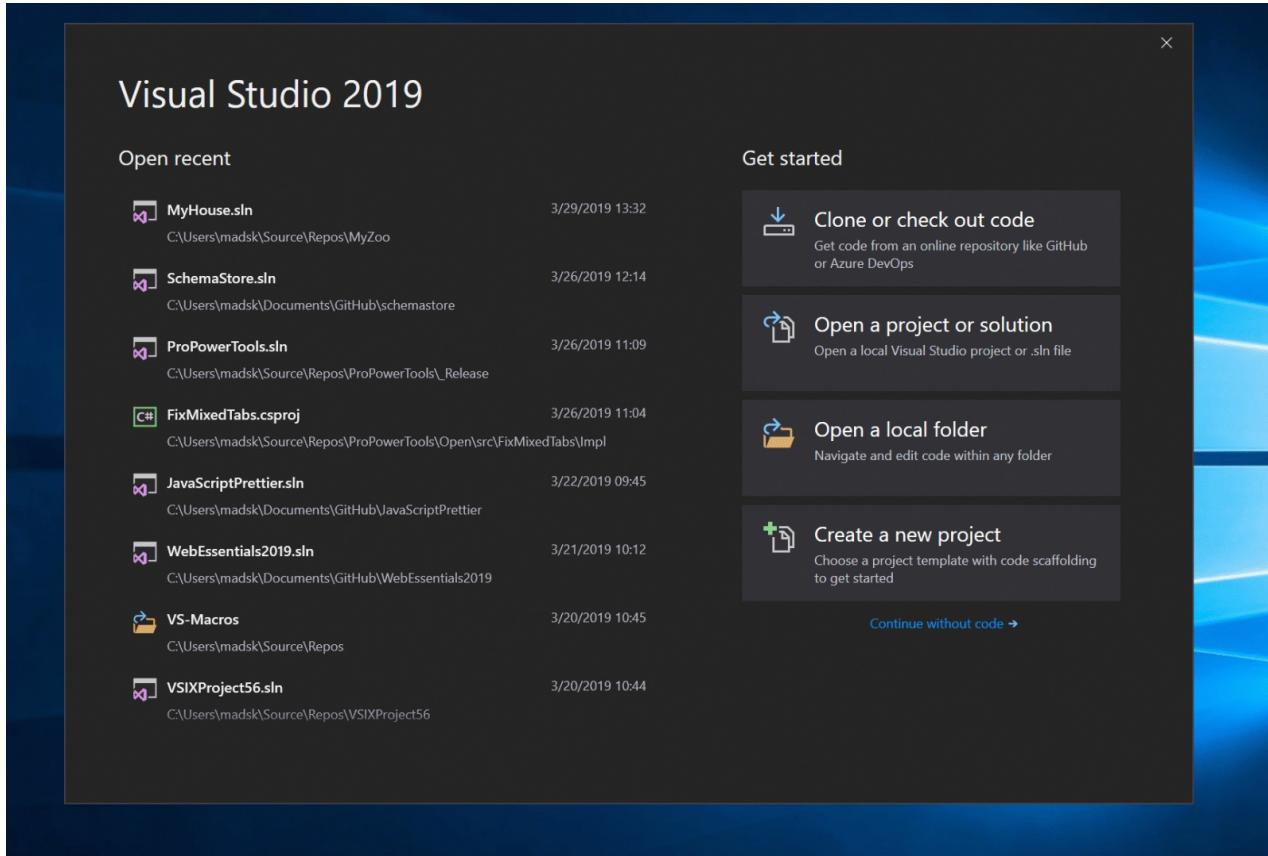
Cloud-first workflow

Something you'll notice when you open Visual Studio 2019 is its new start window.



The start window presents you with several options to get you to code quickly. We've placed the option to clone or

check out code from a repo, first.



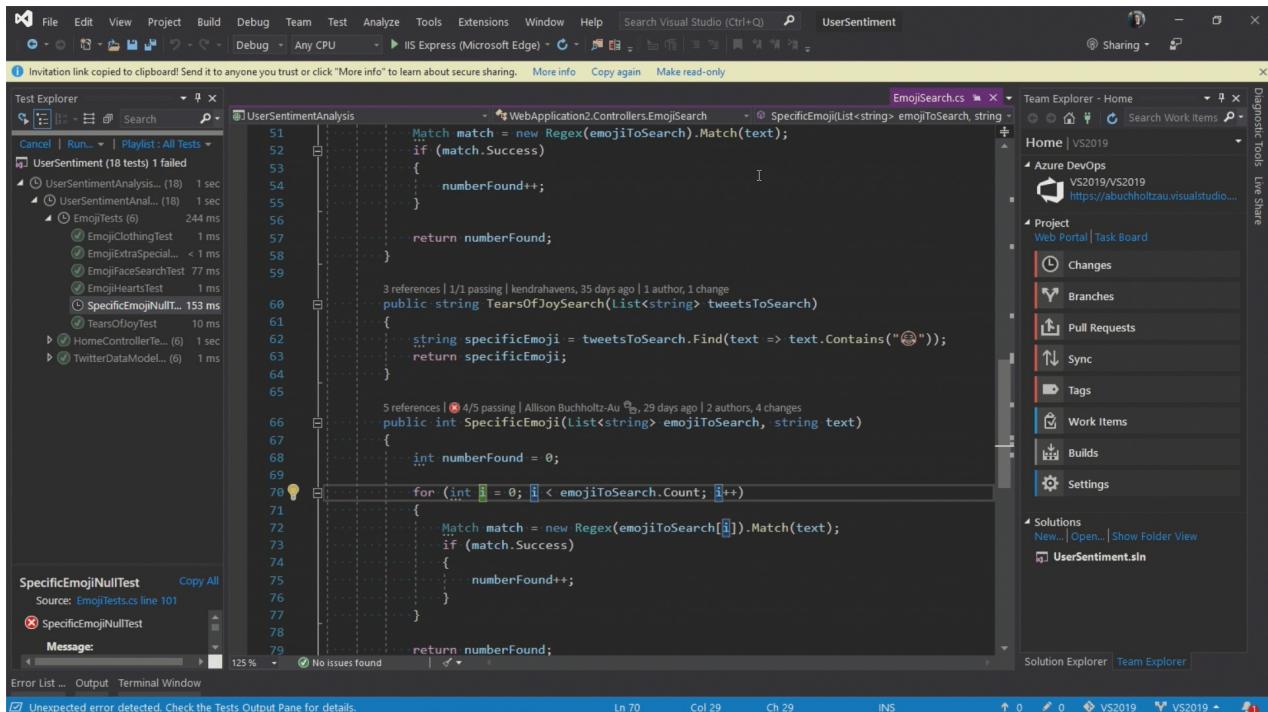
The start window also includes options to open a project or solution, open a local folder, or create a new project.

For more information, see the [Get to code: How we designed the new Visual Studio start window](#) blog post.

Live Share

[Visual Studio Live Share](#) is a developer service that allows you to share a codebase and its context with a teammate and get instant bidirectional collaboration directly from within Visual Studio. With Live Share, a teammate can read, navigate, edit, and debug a project that you've shared with them, and do so seamlessly and securely.

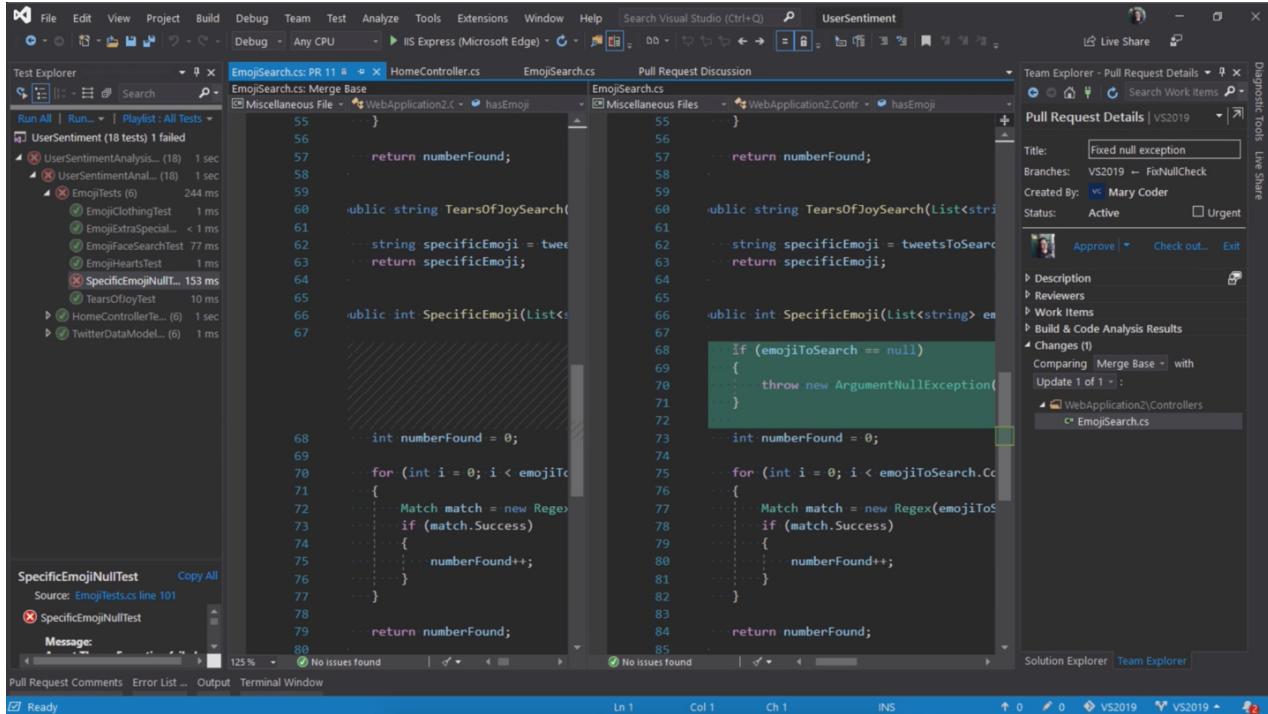
And with Visual Studio 2019, this service is installed by default.



For more information, see the [Visual Studio Live Share for real-time code reviews and interactive education blog post](#) and the [Live Share now included with Visual Studio 2019](#) blog post.

Integrated code reviews

We're introducing a new extension that you can download to use with Visual Studio 2019. With this new extension, you can review, run, and even debug pull requests from your team without leaving Visual Studio. We support code in both GitHub and Azure DevOps repositories.



For more information, see the [Code reviews using the Visual Studio Pull Requests extension](#) blog post.

Debug

View the following video to learn more about how you can zero in with precise targeting while you debug.

Video length: 3.54 minutes

Performance gains

We've taken the once-exclusive C++ data breakpoints and adapted them for .NET Core applications.

The screenshot shows the Visual Studio 2019 interface during a debugging session. The top pane displays the code for `MediaPlayer.cs` with a breakpoint set on line 12. The Locals window (bottom-left) shows variables `args` and `playlist`. The Breakpoints window (bottom-right) shows a data breakpoint on line 12. The status bar at the bottom indicates "Loading symbols for SampleDa...".

Name	Value	Type
args	{string[0x00000000]}	string[]
playlist	{SampleDataBreakpointProject.Playlist}	SampleDataBreakpointProject.Pla...

Name	L	Hit Count
MediaPlayer.cs, line 12 character 13		break alwa...

So whether you're coding in C++ or .NET Core, data breakpoints can be a good alternative to just placing regular breakpoints. Data breakpoints are also great for scenarios such as finding where a global object is being modified or being added or removed from a list.

And, if you're a C++ developer who develops large applications, Visual Studio 2019 has made symbols out of proc, which allows you to debug those applications without experiencing memory-related issues.

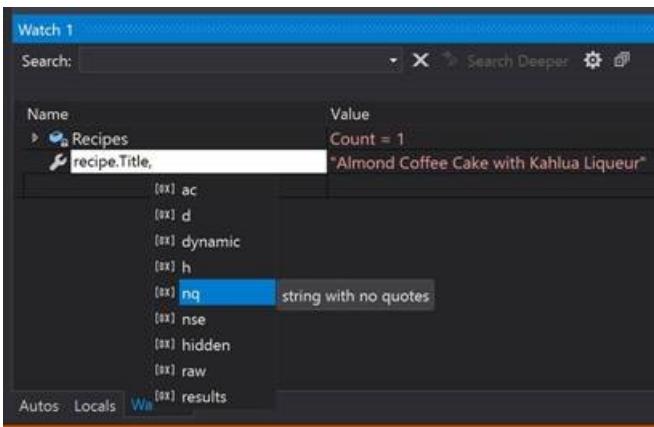
Search while debugging

You've probably been there before, looking in the Watch window for a string amongst a set of values. In Visual Studio 2019, we've added search in the Watch, Locals, and Autos windows to help you find the objects and values you're looking for.

The screenshot shows the Watch 1 window with a search term "Recipes". It displays a single entry: `Recipes` with a value of `Count = 61` and type `System.Collections.Gener...`.

Name	Value	Type
Recipes	Count = 61	System.Collections.Gener...

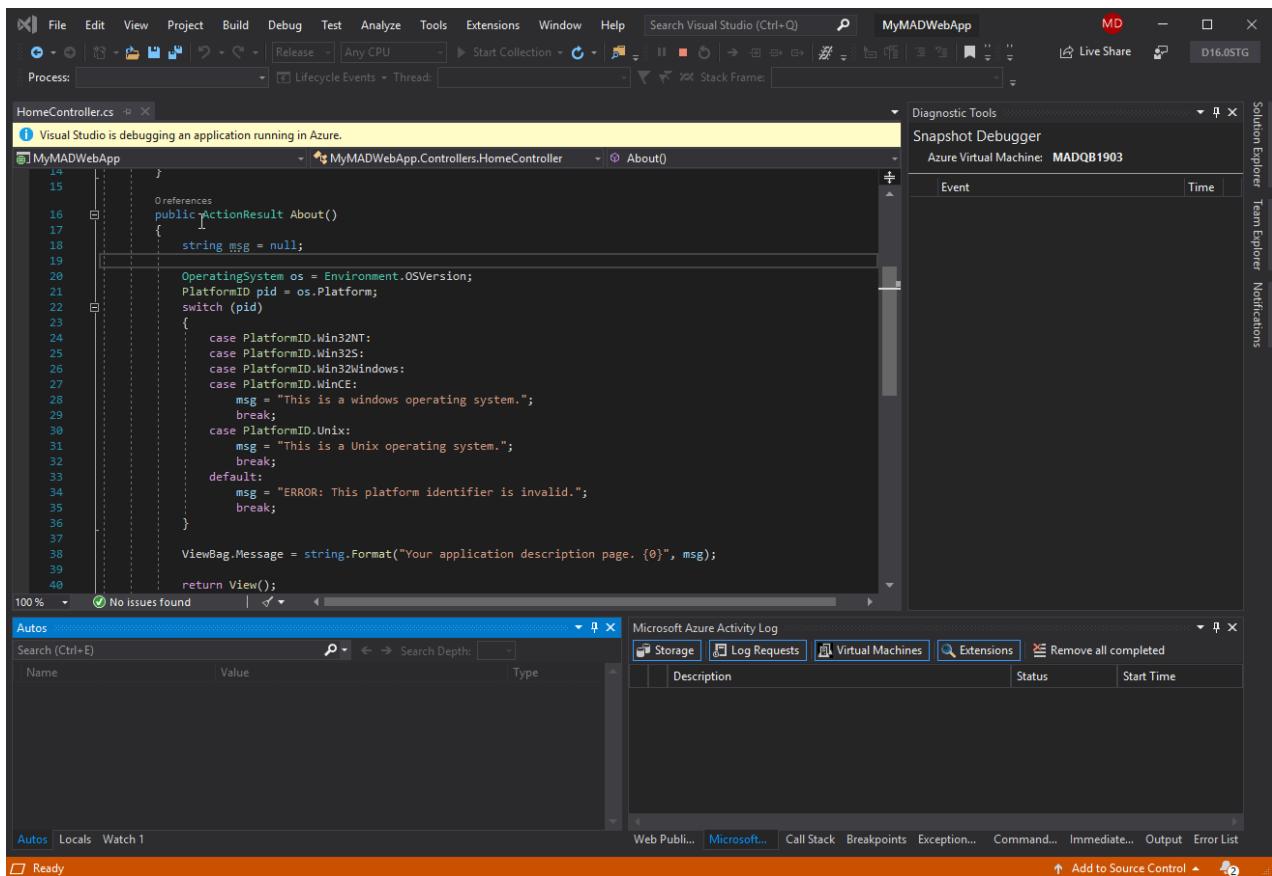
You can also format how a value is displayed within the Watch, Locals, and Autos windows. Double-click one of the items in any of the windows and add a comma (",") to access the drop-down list of possible format specifiers, each of which includes a description of its intended effect.



For more information, see the [Enhanced in Visual Studio 2019: Search for Objects and Properties in the Watch, Autos, and Locals Windows](#) blog post.

Snapshot Debugger

Get a snapshot of your app's execution in the cloud to see exactly what's happening. (This feature is available in Visual Studio Enterprise, only.)

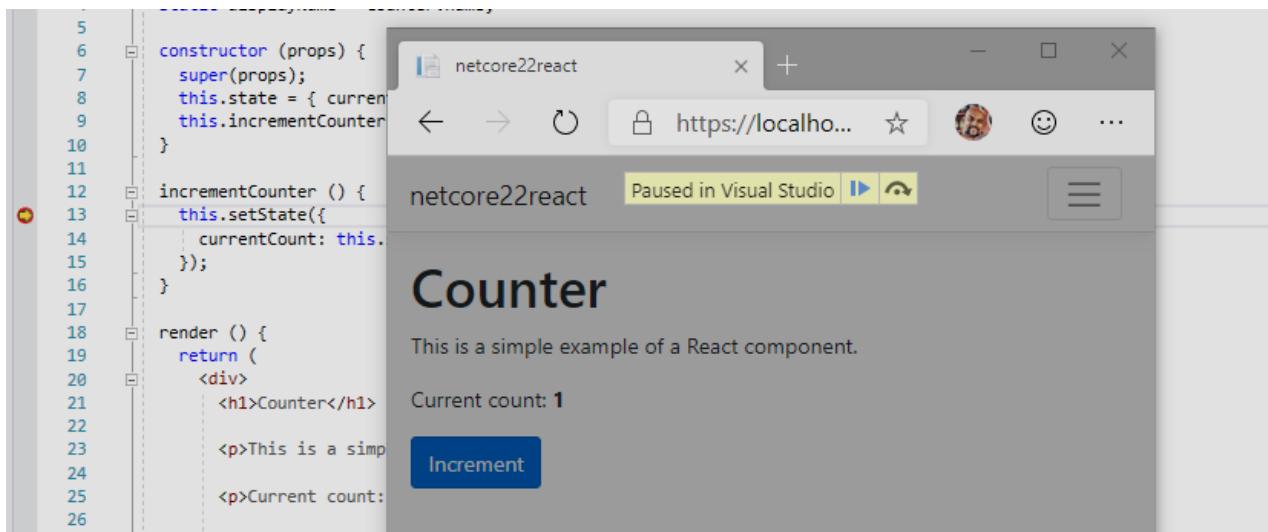


We've added support for targeting ASP.NET (Core and desktop) applications that run on an Azure VM. And, we've added support for applications that run in an Azure Kubernetes Service. The Snapshot Debugger can help you dramatically reduce the time it takes to resolve issues that occur in production environments.

For more information, see the [Debug live ASP.NET Azure apps using the Snapshot Debugger](#) page, and the [Introducing Time Travel Debugging for Visual Studio Enterprise 2019](#) blog post.

Microsoft Edge Insider support

New in 16.2: You can set a breakpoint in a JavaScript application and start a debug session by using the [Microsoft Edge Insider](#) browser. When you do so, Visual Studio opens a new browser window with debugging enabled, which you can then use to step through application JavaScript within Visual Studio.



Pinnable Properties tool

New in 16.4: Now, it's easier to identify objects by their properties while debugging with the new Pinnable Properties tool. Just hover the cursor over a property you want to display in the debugger window of the Watch, Autos, and Locals windows, select the pin icon, and immediately see the information you're looking for at the top of the window!

The Locals window displays a list of variables and their values. A specific entry for 'books' has been pinned, as indicated by a pin icon next to the variable name. The pinned entry is expanded to show a list of nine items, each representing a book object from the 'books' array.

Name	Value	Type
this	{ReadingList.Models.BookManager}	ReadingList.Mod...
books	Count = 100	System.Collections...
[0]	{ReadingList.Models.Book}	ReadingList.Mod...
[1]	{ReadingList.Models.Book}	ReadingList.Mod...
[2]	{ReadingList.Models.Book}	ReadingList.Mod...
[3]	{ReadingList.Models.Book}	ReadingList.Mod...
[4]	{ReadingList.Models.Book}	ReadingList.Mod...
[5]	{ReadingList.Models.Book}	ReadingList.Mod...
[6]	{ReadingList.Models.Book}	ReadingList.Mod...
[7]	{ReadingList.Models.Book}	ReadingList.Mod...
[8]	{ReadingList.Models.Book}	ReadingList.Mod...

What's next

We update Visual Studio 2019 often with new features that can make your development experience even better. To learn more about our latest innovations, check out the [Visual Studio Blog](#). And for a record of what we've released in preview to date, take a look at the [Preview Release Notes](#).

Want to know more about what else is in the works for Visual Studio 2019? See the [Visual Studio Roadmap](#).

Give us feedback

Why send feedback to the Visual Studio team? Because we take customer feedback seriously. It drives much of what we do.

- If you want to make a suggestion about how we can improve Visual Studio, you can do so by using the [Suggest a Feature](#) tool.
- If you experience a hang, crash, or other performance issue, you can easily share repro steps and supporting files with us by using the [Report a Problem](#) tool.

See also

- [Visual Studio 2019 release notes](#)
- [What's New in the Visual Studio 2019 SDK](#)
- [Visual Studio 2019 for Mac release notes](#)
- [Microsoft Build 2019 conference](#)
- [Microsoft Connect\(\); 2018 conference](#)

Visual Studio Customer Experience Improvement Program

9/30/2019 • 2 minutes to read • [Edit Online](#)

The Visual Studio Customer Experience Improvement Program (VSCEIP) is designed to help Microsoft improve Visual Studio over time. This program [collects information about errors](#), computer hardware, and how people use Visual Studio, without interrupting users in their tasks at the computer. The information that's collected helps Microsoft identify which features to improve. This document covers how to opt in or out of the VSCEIP.

NOTE

If you're interested in viewing or deleting personal data, please review Microsoft's guidance at [Windows Data Subject Requests for the GDPR](#). If you're looking for general information about GDPR, see the GDPR section of the [Service Trust Portal](#).

NOTE

VSCEIP telemetry opt in or out settings do not apply to 'Report a Problem' in Visual Studio. When you report a problem logs are collected and sent to Microsoft only when you provide permission by clicking 'Submit'. If you are interested in managing logs before submitting to 'Report a Problem' please see [Feedback Data Privacy](#) for more details.

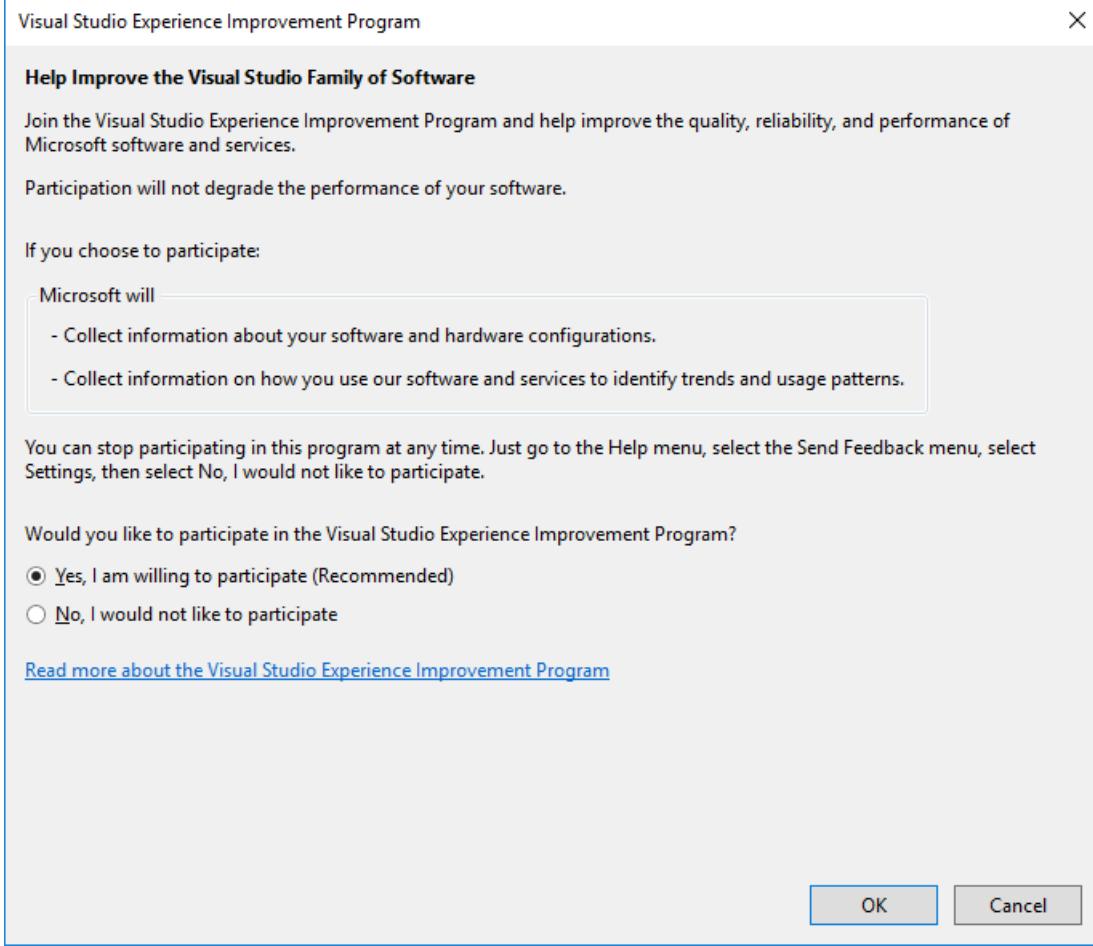
Opt in or out

The VSCEIP is turned on by default. You can turn it off, or back on again, by following these instructions:

1. In Visual Studio, choose **Help > Send Feedback**, and then select **Settings**.

The **Visual Studio Experience Improvement Program** dialog box opens.

2. To opt out, select **No, I would not like to participate**, and then select **OK**. To opt in, select **Yes, I am willing to participate**, and then select **OK**.



Registry settings

If you install the [Build Tools for Visual Studio](#), you must update the registry to configure the VSCEIP. Enterprise customers can construct a group policy to opt in or out of the VSCEIP by setting a registry-based policy.

The relevant registry key and settings are as follows:

- On a 64-bit OS, Key =
HKEY_LOCAL_MACHINE\SOFTWARE\Wow6432Node\Microsoft\VSCommon\15.0\SQM
- On a 32-bit OS, Key = **HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\VSCommon\15.0\SQM**
- When Group Policy is enabled, Key =
HKEY_LOCAL_MACHINE\Software\Policies\Microsoft\VisualStudio\SQM
- On a 64-bit OS, Key =
HKEY_LOCAL_MACHINE\SOFTWARE\Wow6432Node\Microsoft\VSCommon\16.0\SQM
- On a 32-bit OS, Key = **HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\VSCommon\16.0\SQM**
- When Group Policy is enabled, Key =
HKEY_LOCAL_MACHINE\Software\Policies\Microsoft\VisualStudio\SQM

Entry = **OptIn**

Value = (DWORD)

- **0** is opted out (turn off the VSCEIP)
- **1** is opted in (turn on the VSCEIP)

Caution

Incorrectly editing the registry may severely damage your system. Before making changes to the registry, you should back up any valued data on the computer. You can also use the **Last Known Good Configuration** startup option if you encounter problems after manual changes have been applied.

For more information about the information collected, processed, or transmitted by the VSCEIP, see the [Microsoft Privacy Statement](#).

See also

- [Diagnostic information collected by Visual Studio](#)
- [Visual Studio feedback options](#)
- [How to report a problem with Visual Studio](#)
- [Visual Studio Developer Community](#)
- [Microsoft Privacy Statement](#)

System-generated logs collected by Visual Studio

10/18/2019 • 3 minutes to read • [Edit Online](#)

Visual Studio collects system-generated logs to fix problems and improve the quality of the product through the [Visual Studio Customer Experience Improvement Program](#). This article provides some information about the types of data we collect and how we use it. It also provides tips on how extension authors can avoid inadvertent disclosure of personal or sensitive information.

Types of collected data

Visual Studio collects system-generated logs for crashes, hangs, UI unresponsiveness, and high CPU or memory usage. We also collect information about errors encountered during product installation or usage. The collected data varies based on the error, and may include stack traces, memory dumps, and exception information:

- For high CPU usage and unresponsiveness, stack traces of relevant Visual Studio threads are collected.
- For cases where stack traces of some threads aren't enough to determine the root cause of the issue, for example, crashes, hangs or high memory usage, we collect a memory *dump*. The dump represents the state of the process when the error occurred.
- For unexpected error conditions, for example, an exception while trying to write to a file on disk, we collect information about the exception. The information includes the name of the exception, the stack trace of the thread where the exception occurred, the message associated with the exception, and other information relevant to the specific exception.

The following example of collected data shows an exception name, stack trace, and exception message:

```
"Reserved.DataModel.Fault.Exception.TypeString": "System.IO.IOException",
"Reserved.DataModel.Fault.Exception.StackTrace": "System.IO.__Error.WinIOError(Int32,String)\r\n
System.IO.FileStream.Init(String, FileMode, FileAccess, Int32, Boolean, FileShare, Int32, FileOptions, SECURITY_ATTRIBUTES, String, Boolean, Boolean, Boolean)\r\n
System.IO.FileStream..ctor(String, FileMode, FileAccess, FileShare, Int32, FileOptions, String, Boolean, Boolean, Boolean)\r\n
System.IO.StreamWriter.CreateFile(String, Boolean, Boolean)\r\n
System.IO.StreamWriter..ctor(String, Boolean, Encoding, Int32, Boolean)\r\n
System.IO.StreamWriter..ctor(String, Boolean)\r\n
System.IO.File.CreateText(String)\r\n
Microsoft.VisualStudio.Setup.Services.FileSystem.CreateText(String, Boolean)\r\n
Microsoft.VisualStudio.Setup.Cache.ChannelManifestRepository.WriteChannelManifest(IChannelManifest, String, String)\r\n
Microsoft.VisualStudio.Setup.Cache.ChannelManifestRepository.AddChannel(ChannelManifestPair, Boolean)\r\n
Microsoft.VisualStudio.Setup.Cache.CacheManager.AddChannel(ChannelManifestPair, Boolean)\r\n
Microsoft.VisualStudio.Setup.ChannelManager.\u2026<UpdateAsync>d_37.MoveNext()\r\n",
"Reserved.DataModel.Fault.Exception.Message": " The process cannot access the file 'C:\\\\Users\\\\[UserName]\\\\AppData\\\\Local\\\\Microsoft\\\\VisualStudio\\\\Packages\\\\_Channels\\\\4CB340F5\\\\channelManifest.json' because it is being used by another process."
```

How we use system-generated logs

The workflow to determine the root cause of an error varies depending on the type of error and its severity.

Error classification

Based on the logs, errors are classified and counted to prioritize their investigation. For example, we may discover that "System.IO.__Error.WinIOError" at "System.IO.FileStream.Init" has occurred 500 times in version <x> of the

product, and has the highest rate of occurrence in that version.

Work items for tracking

Work items for individual, prioritized errors are created and assigned to engineers for investigation. These work items typically contain the classification, priority, and diagnostic information relevant to the type of error. This information is derived from the collected system-generated logs for the error. For example, a work item for a crash might contain the stack trace where the crash is occurring.

Error investigation

Engineers use the information available in a work item to determine the root cause of an error. In some cases, they need more information than what's present in the work item, in which case they refer to the original system-generated log that was collected. For example, an engineer might inspect a memory dump to understand a product crash.

Tips for extension authors

Extension authors should limit exposure of personal information by not using personal or other sensitive information in the names of their modules, types, and methods. If a crash or similar error condition occurs with that code on the stack, that information gets collected as part of the system-generated logs.

Opt out of data collection

Given the purpose of the data we collect and the constraints on its access and retention, we recommend that you use the default privacy settings for Visual Studio and Windows. However, you can [opt out](#) of the Visual Studio Experience Improvement Program. To opt out of system-generated log collection for all programs, see [Diagnostics, feedback, and privacy in Windows 10](#). Options may vary depending on the version of Windows you're using.

See also

- [Visual Studio Customer Experience Improvement Program](#)
- [Diagnostics, feedback, and privacy in Windows 10](#)

Resources for troubleshooting integrated development environment errors

10/25/2019 • 2 minutes to read • [Edit Online](#)

Not all error messages have a specific associated Help topic. If the information in the error message does not help you resolve the problem, you can consult other resources such as Knowledge Base articles, forums, or product support.

NOTE

This topic applies to Visual Studio on Windows. For Visual Studio for Mac, see [Troubleshoot Visual Studio for Mac](#).

Knowledge Base articles

You can search the Knowledge Base (KB) online for articles about product issues. Not all issues have a corresponding KB article, but errors encountered by a significant number of customers are typically documented. You can access KB articles on the [Microsoft Support](#) website.

Developer forums

Forums let you interact with other developers, and also Microsoft employees. If you encounter an error that you cannot find a resolution for, you can post questions about the issue on a forum. You can also search the newsgroups to see whether others have posted about the same issue.

You can access forums, blogs, chats, and other resources on the [Microsoft Technical Communities](#) website.

Product support

If you still have questions after you try the other resources, you can contact Microsoft support services by visiting the [Microsoft Support](#) website. For information about product support available in your area, see the [Visual Studio feedback options](#) page.

See also

- [Troubleshoot network-related or proxy errors](#)
- [Troubleshooting \(Visual Studio for Mac\)](#)

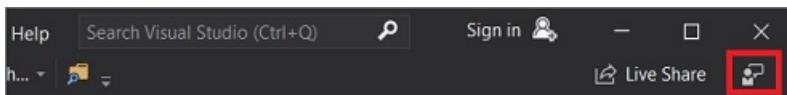
Visual Studio feedback options

12/5/2019 • 2 minutes to read • [Edit Online](#)

Why send feedback to us? Because we take customer feedback seriously; it drives much of what we do. Here's how to contact us so that we can route your feedback to the right person or team.

Report a problem

If you experience problems with Visual Studio—such as crashes, sluggish performance, or unexpected behavior—let us know by using the **Report a Problem** tool. In Visual Studio, choose the feedback icon in the upper-right corner, or choose **Help > Send Feedback > Report a Problem** from the menu bar.



Next, Visual Studio opens the [Developer Community](#) interface, where you can sign in to report the problem. For more information, see [How to report a problem with Visual Studio](#).

If you experience problems with Visual Studio—such as crashes, sluggish performance, or unexpected behavior—let us know by using the **Report a Problem** tool. In Visual Studio, choose the feedback icon next to the search box, or choose **Help > Send Feedback > Report a Problem** from the menu bar.



Next, Visual Studio opens the [Developer Community](#) interface, where you can sign in to report the problem. For more information, see [How to report a problem with Visual Studio](#).

Suggest a feature

If you have an idea or a suggestion to make Visual Studio better, let us know by using the **Suggest a Feature** tool. In Visual Studio, choose the feedback icon in the upper-right corner, or choose **Help > Send Feedback > Suggest a Feature** from the menu bar.

Next, Visual Studio opens the [Developer Community](#) interface, where you can sign in to share your idea. For more information, see [Suggest a feature for Visual Studio](#).

If you have an idea or a suggestion to make Visual Studio better, let us know by using the **Provide a Suggestion** tool. In Visual Studio, choose the feedback icon next to the search box, or choose **Help > Send Feedback > Provide a Suggestion** from the menu bar.

Next, Visual Studio opens the [Developer Community](#) page, where you can sign in to share your idea. For more information, see [Suggest a feature for Visual Studio](#).

Improve the documentation

There are two ways that you can help us improve the documentation:

- Use the **Is this page helpful?** response tool at the top-right of any documentation page.
- Use your GitHub account to provide feedback for any Visual Studio page on docs.microsoft.com. To do so, choose the **Send feedback about > This page** button at the bottom of any documentation page.

Contact Microsoft support

For Visual Studio support information, see the [Product life cycle & servicing](#) page. For other Microsoft products and services, see [Microsoft support](#) for online help.

NOTE

Support outside the United States and Canada may vary. For a list of regional contacts, see [Microsoft worldwide sites](#).

For larger organizations that require managed support directly from Microsoft, contracts are available through various Enterprise Support offerings. For more information, see [Microsoft Enterprise Support solutions](#).

If the product came installed with a new computer or device, the hardware manufacturer provides technical support and assistance for this software. Contact the manufacturer directly for support.

Microsoft support services are subject to then-current prices, terms, and conditions. Prices, terms, and conditions can change without notice.

Ask the community

If you want to share questions and answers with other developers, consider connecting with them on the following community sites:

- [MSDN forums](#)
- Visual Studio on [Reddit](#)
- [Stack Overflow](#)

You can also view code from other developers and share your own examples on the [Browse code samples](#) page.

See also

- [Troubleshoot installation and upgrade issues](#)
- [Developer Community data privacy](#)

How to report a problem with Visual Studio or Visual Studio Installer

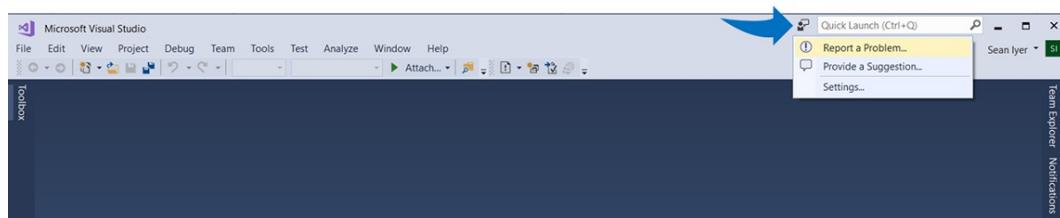
4/26/2019 • 4 minutes to read • [Edit Online](#)

NOTE

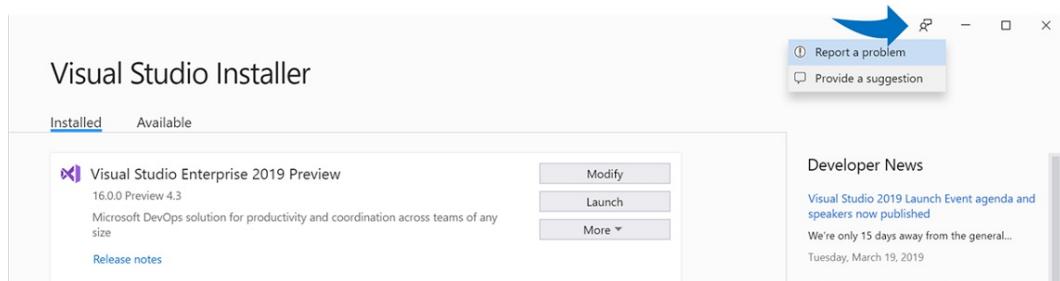
For Visual Studio for Mac, see [How to report a problem in Visual Studio for Mac](#).

You can report a problem from either Visual Studio or its installer by using the Feedback Tool included in them. The Feedback Tool enables you to easily include diagnostic information in your feedback and helps Visual Studio teams to diagnose and fix issues much more effectively. Here are the steps to report a problem.

1. In **Visual Studio**, select the feedback icon in the upper-right corner and select Report a Problem. You can also access the feedback tool from the menu **Help > Send Feedback > Report a Problem**.



Alternatively, report a problem in **Visual Studio Installer** if you can't install Visual Studio or are unable to access the feedback tool within Visual Studio. In the Installer, select the feedback icon in the upper-right corner and select Report a Problem.



2. If not signed in, select **Sign In** as shown in the following screenshot. Follow the instructions on-screen to sign in.



Not only can you report a problem when you are signed in, but you can also vote and comment on any existing feedback.

3. Once signed in, you will be able to see your **Problems** and **Activity** on the **Items I follow** screen

The screenshot shows the 'Developer Community' section of the Visual Studio Feedback website. At the top, there's a search bar with the placeholder 'Search for anything'. On the right, there's a user profile for 'Maria Ghiondea' with a small photo. Below the header, the title 'Items I Follow' is displayed. Underneath, a list of reported issues is shown:

- html list remain in front while debugging**
Reported by Marc Roussel on 2/16/2018, 10:57:09 AM.
1 vote. Status: **Need More Info**. Tags: Windows 10.0, Visual Studio 2017 version 15.5, Debugger, JavaScript.
Description: When we hit a breakpoint in Javascript file after selection an option in a select list, this list remain in front of the screen regardless where we are even in this visual studio feedback the list is front of it. I don't...
Last updated by Marc Roussel on 5/11/2018.
- Xamarin Android project cannot build because R\$anim.class is in use**
Reported by Bernard Davison on 11/26/2017, 11:06:01 AM.
36 votes. Status: **Closed - Fixed**. Tags: windows 6.1, visual studio 2017 version 15.5 preview, Known Issue in: Visual Studio 2017 Version 15.5.
Description: While trying to build a deploy a Xamarin Android project I'm continually getting deployment errors on the first build. Subsequent builds are failing with Severity Code Description Project File Line Suppression...
Last updated by Maria Ghiondea [MSFT] on 3/8/2018.

At the bottom of the list, there's a message: 'Can't find an existing problem that meets your criteria?' followed by a blue button labeled 'Report new problem'.

4. Visual Studio provides an interface to search for your problem and see if others have reported it. If someone has reported it, "up-vote" it to let us know.

NOTE

In order to search, please input the desired text into the search box and either click Enter or press the Search icon.

The screenshot shows the 'Developer Community' section of the Visual Studio Feedback website with a search query 'Visual Studio' entered into the search bar. The search results are displayed:

- visual studio**
Reported by Burak Gokoglu on 4/13/2017, 2:28:10 PM.
0 votes. Status: **Closed - Fixed**. Tags: windows 6.1, visual studio 2017 (version 15.1), fixed-in: visual studio 2017 (version 15.2).
Description: upon opening new solution vs crashes after last update.
Last updated by John Catenzaro - MSFT [MSFT] on 5/10/2017.
- Visual Studio not Visual**
Reported by Ray Hall on 4/17/2018, 2:56:11 AM.
0 votes. Status: **Closed - Not a Bug**. Tags: Windows 6.1, Visual Studio 2017 version 15.6, setup, C++.
Description: I have used Embarcadero C++ Builder for many years for my projects. I decided to take a look at **Visual Studio**. I found it is not **visual** the way I expected. There is no GUI tools and it is all code based. Very...
Last updated by Jeff Kelly [MSFT] on 4/17/2018.
- VISUAL STUDIO 2017**
Reported by Rafael FEBRES on 2/18/2018, 5:52:25 PM.
0 votes. Status: **Closed - Not Enough Info**. Tags: windows 10.0, visual studio 2017 version 15.5.
Description: I DOWNLOADED VISUAL STUDIO 2017 FROM THE 2017 IDE IS NOT INCLUDED. I TRIED TO FIND THE IDE TO...

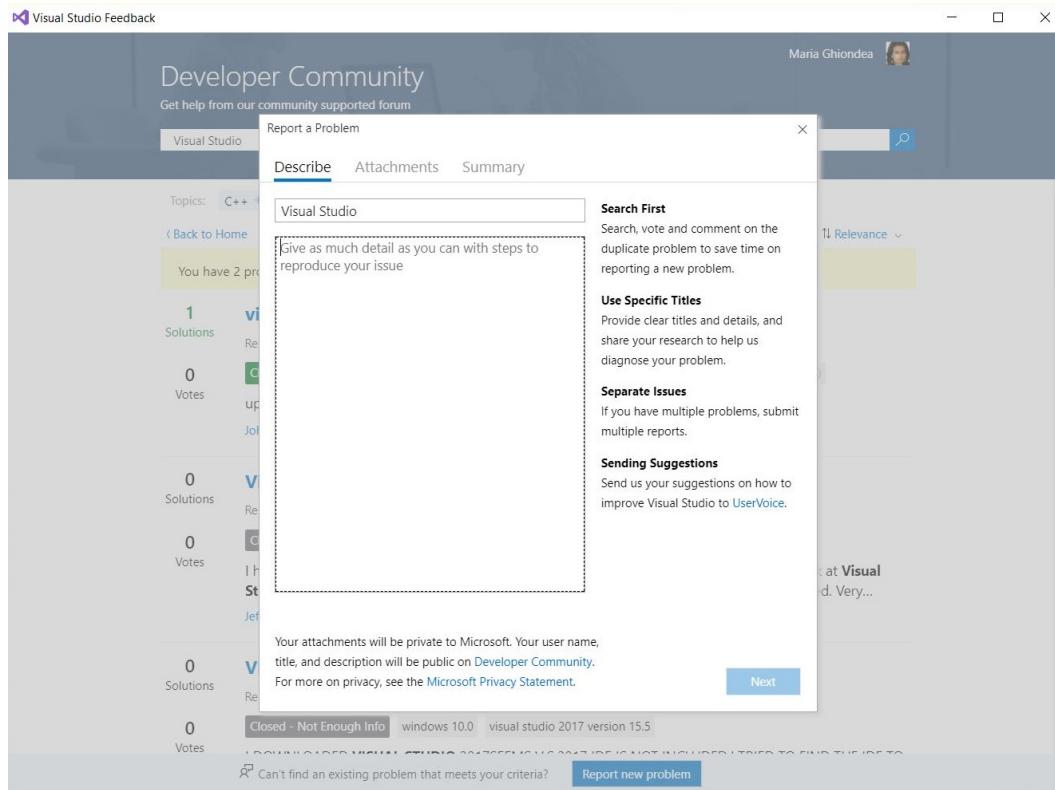
At the bottom of the list, there's a message: 'Can't find an existing problem that meets your criteria?' followed by a blue button labeled 'Report new problem'.

5. If you don't find the problem you encountered, choose **Report new problem** at the bottom of the screen.

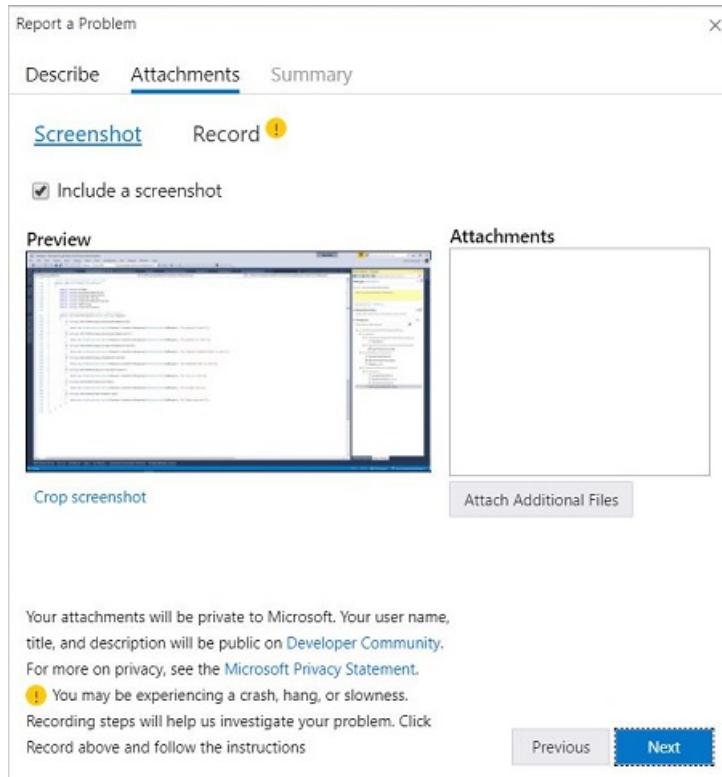
NOTE

The **Report new problem** button only appears in the Visual Studio interface for Developer Community. You can't report a problem directly on the [Developer Community](#) website.

6. Create a descriptive title for the problem that helps us route it to the correct Visual Studio team.
7. Give us any additional details, and if possible, provide us with the steps to reproduce the problem.



8. Select **Next** to move to the **Attachments** tab. Here, you can capture your current screen to send it to Microsoft. To attach additional screenshots or other files, choose **Attach Additional Files**.



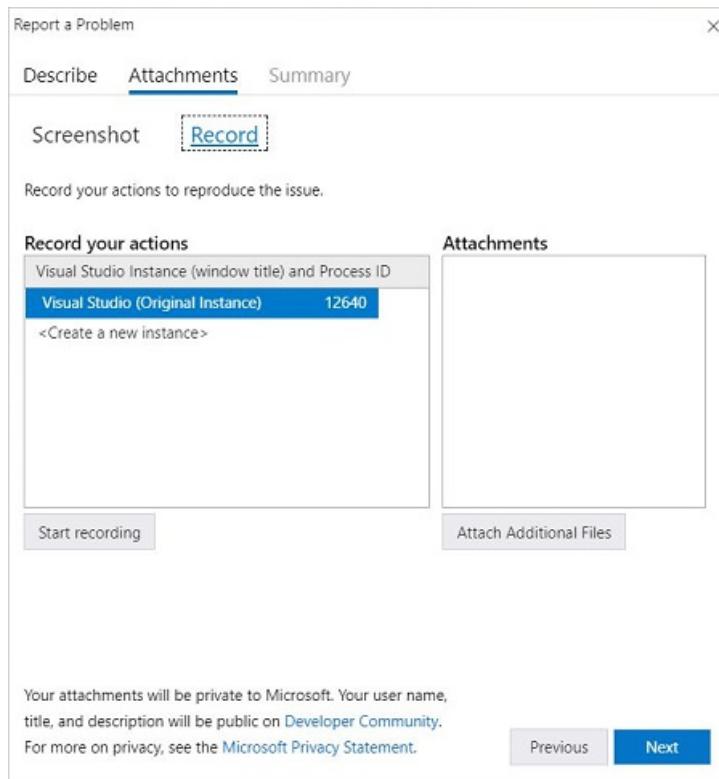
9. If you don't want to attach a screenshot or [record a repro](#), select **Next** to move to the **Summary** tab.
10. Select **Submit** to send your report, along with any images and trace or dump files. (If the **Submit** button is grayed out, make sure that you've provided a title and description for the report.)

For information about what data is collected, see [Data we collect](#).

Record a repro

Trace and heap dump files are useful in helping us diagnose problems. We appreciate it when you use the **Report a Problem** tool to record your repro steps and send the data to Microsoft. Here's how to do so:

1. After you enter a title and description for your problem, select **Next** to move to the **Attachments** tab.
2. Select the **Record** tab.
3. Under **Record your actions**, select the current instance of Visual Studio if you can reproduce the problem there. If you can't, for example if Visual Studio is hung, select **<Create a new instance>** to record the actions in a new instance of Visual Studio.
4. Select **Start Recording**. Give permission to run the tool.



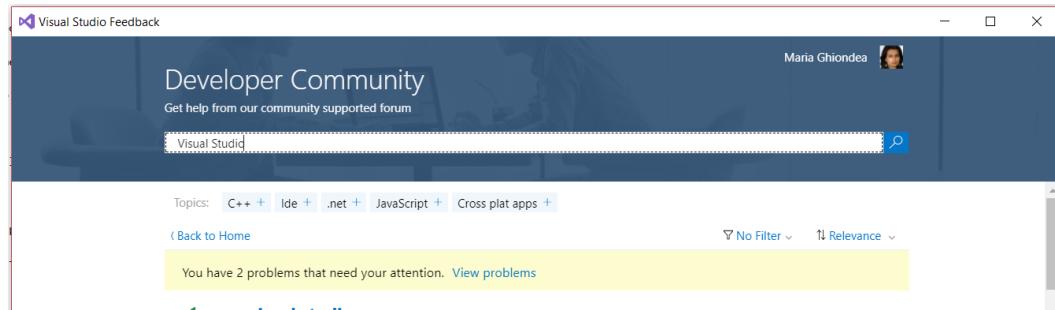
5. When the **Steps Recorder** tool appears, perform the steps that reproduce the problem.
6. When you're done, choose the **Stop Record** button.
7. Wait a few minutes for Visual Studio to collect and package the information that you recorded.

For information about what data is collected, see [Data we collect](#).

When further information is needed (Need More Info)

Starting in Visual Studio 2017 Version 15.5, there's a new workflow to help users provide additional information about problem reports.

1. When a Microsoft engineer sets the [Visual Studio Developer Community](#) problem to the **Need More Info** state, any user that posted, voted, followed, or commented on the problem gets a notification in the **Report A Problem** tool in Visual Studio.



2. Click on the **View Problems** link to filter and sort the view to the problems that need attention. These problems also have an indicator next to them, to differentiate them in general search.
3. Click on a problem to see the problem details view.

The screenshot shows a Microsoft Edge browser window titled "Visual Studio Feedback". The main content area is titled "Developer Community" with the sub-instruction "Get help from our community supported forum". A search bar at the top right contains the placeholder "Search for anything". Below the search bar is a yellow banner with the text "The Visual Studio team needs more information about this problem. [View their request and respond](#)". A post is displayed with the title "html list remain in front while debugging". It has a vote count of 1, reported by Marc Roussel on 2/16/2018, 10:57:09 AM. The post includes the "Need More Info" button, which is highlighted in red, and tags for Windows 10.0, Visual Studio 2017 version 15.5, Debugger, and JavaScript. The post content describes a bug where a list remains in front of the screen when a breakpoint is hit in a JavaScript file after selecting an option in a select list.

4. To view the **Need More Info** request, click the **View their request and respond** link in the problem details view. A dialog box shows the request.

The screenshot shows a modal dialog box titled "Need More Info". The tabs at the top are "Request" (which is selected), "Comment", "Attachments", and "Summary". The main content area contains a dashed box with the text "The Visual Studio team needs more information about this problem. Please view the request below and respond with the appropriate information." Below this, a message from "Joe Davis (WEB TOOLS) [MSFT] on 2/27/2018" asks for information about the browser being used. At the bottom, there is a note about attachments being private to Microsoft and user names and comments being public on the Developer Community. A "Next" button is visible on the right.

5. You can provide more information by adding comments, attachments, or recording steps. This experience is similar to reporting a new problem or providing additional information when voting on a problem.
6. The requesting Microsoft engineer receives a notification about the extra information provided. If they have enough information to investigate, the problem state changes. Otherwise, the engineer asks for even further information.

NOTE

- When you reply, the notification goes away. In its place, you see a banner that thanks you and facilitates a way to provide even more information.
- Once the issue changes state, the notification goes away for everyone that's following the issue.
- More than one person can reply on the same **Need More Info** request.
- There isn't a **Need More Info** workflow on [Developer Community](#) when you access it directly through a web browser, but you can also provide comments and attachments there.

Search for solutions or provide feedback

If you don't want to or can't use Visual Studio to report a problem, there's a chance the problem has already been reported and a solution posted on the [Visual Studio Developer Community](#) page.

If you don't have a problem to report but want to suggest a feature, there's a place for that, too. For more information, see the [Suggest a feature](#) page.

See also

- [Visual Studio feedback options](#)
- [Report a problem with Visual Studio for Mac](#)
- [Report a problem with C++](#)
- [Visual Studio Developer Community](#)
- [Developer Community data privacy](#)

Report a problem: States and FAQ

12/5/2019 • 4 minutes to read • [Edit Online](#)

The Report a Problem tool enables the Visual Studio developer community to submit issues. Each one of your problem reports becomes a work item in our core engineering system, empowering you to engage directly with our product teams to help us identify and resolve impactful issues. Your feedback submitted with rich diagnostic information is critical to improving the Visual Studio product family. We really appreciate you taking the time to report problems.

In addition, you can vote on feedback from other community members to bring more attention to a problem and help fix it faster.

Problem status

After you report a problem, states indicate where your submissions are in their lifecycle. As Microsoft teams review your feedback, they set it with an appropriate state. Track the progress of your problem reports by referencing the states listed below, along with their meaning and color indicators.

New

New indicates that the bug or issue is newly reported, and no action has been taken on it yet.

Triaged

Triaged indicates that preliminary steps such as moderation, translation, and initial check for duplicates are complete. Your ticket has been routed to the appropriate engineering team for consideration.

Under Consideration

Under Consideration indicates that Microsoft is reviewing your problem for community impact and will prioritize it accordingly. If the community impact isn't clear or significant yet, we'll continue to monitor the problem in this state.

Under Investigation

Under Investigation indicates that engineers are actively investigating your problem to find a resolution.

Need More Info

Need More Info indicates that we need more diagnostic information from you so that we can go forward with the investigation. [Learn how to respond to Need More Info requests.](#)

Fixed - Pending Release

Fixed - Pending Release indicates that we have a fix for the problem and it will be available in an upcoming preview or release. When the fix becomes available in a preview, the problem is tagged with a 'fixed in' tag specifying the preview version.

Closed - Fixed

Closed - Fixed indicates that we've released a fix for the problem. The problem is also now tagged with a "fixed

in:" tag specifying the release version.

Closed - Duplicate

Closed - Duplicate indicates that your issue has already been reported via another feedback. We'll provide you with the link where you can track the original problem report.

Closed - Lower Priority

Closed - Lower Priority To focus on bringing each of you in our developer community the best value, we prioritize issues with the highest customer impact. Although we're unable to address this particular issue at this time, be assured that all your feedback is valuable and helps improve Visual Studio.

Closed - Not a Bug

Closed - Not a Bug indicates that we've determined that the reported functionality is by current design.

Closed - Not Enough Info

Closed - Not Enough Info indicates that we don't have enough information to investigate this for you. We'll be happy to reconsider the feedback after the necessary information is available.

Closed - Other Product

Closed - Other Product indicates we've determined that your issue applies to another product. See the comment from Microsoft for which external product and any related links.

Closed - Won't Fix

Closed - Won't Fix indicates that we aren't pursuing this issue due to factors such as lack of product direction alignment or community impact. See the comment from Microsoft for any additional clarity. Although we're unable to address this particular issue, be assured that all your feedback is valuable and helps improve Visual Studio.

FAQ

How can I increase the chance of my problem getting resolved quickly?

We recommend using search to ensure that the problem you're about to report hasn't already been reported. If you find an existing item matching your problem, follow and vote on that problem ticket.

Provide all the information you can to help our teams reproduce what you're experiencing. This information includes necessary repro steps, code fragments, screenshots, repro recordings, log files, and other artifacts. Here is [how to report a problem in Visual Studio](#).

How is my feedback prioritized?

We receive a large number of valuable problems from our customers. To ensure that we're bringing the best value to each of you in our developer community, we prioritize action on feedback that has the highest community impact.

If we aren't able to respond personally to your feedback, know that we fully appreciate your input. Be assured that all your feedback gets to the right team.

We truly value the time you invest in making Visual Studio better.

What actions can I take if I'm not satisfied with the resolution?

Our teams do their best to diagnose and fix any issues you experience, however there may be times when you're not fully satisfied with our recommendation. Comment back on the feedback and let us know exactly what you're not satisfied with, and we'll try our best to ensure that we meet your needs.

How will I get notified of progress on my feedback?

Microsoft engineering teams will communicate with you by commenting on the feedback ticket and changing the state of your ticket as they make progress. Watch for e-mail notifications that are sent when ticket state changes or a comment is posted. You can manage frequency of notifications in Profile and Preferences settings on Developer Community site.

Why can't I add a problem for Visual Studio IDE on the Developer Community website?

Reporting a problem through Visual Studio allows for diagnostic information to automatically be included in the report. It's essential information that gives our engineers the context they need to fully understand your issue and work to resolve it.

When you report through Visual Studio, you can easily share rich diagnostic information with us, such as large log files, crash information, screenshots, repro recording, and other artifacts that help us deliver higher-quality resolutions faster to you.

How to increase the chances of a performance issue being fixed

12/7/2019 • 8 minutes to read • [Edit Online](#)

The "Report a problem" tool is widely used by Visual Studio users to report a range of problems. The Visual Studio team spots crash and slowness trends in user feedback and addresses issues impacting a broad swath of users. The more actionable a specific feedback ticket is, the more likely it will be diagnosed and resolved quickly by the product team. This document describes the best practices while reporting crash or slowness issues to make them more actionable.

General best practices

Visual Studio is a large, complex platform that supports a multitude of languages, project types, platforms, and more. How it performs is a function of which components are installed and active in a session, the extensions installed, the Visual Studio settings, machine configuration, and finally the shape of the code that is being edited. Given the number of variables, it is hard to tell whether the problem report from one user has the same underlying issue as a problem report from another user, even though the visible symptom is the same. Given that, here are some best practices to ensure your specific problem report has higher likelihood of being diagnosed.

Provide as specific a title as possible

Look for distinct signatures for the problem being reported and include as much as possible in the title. If the title is descriptive, it is less likely that users with unrelated problems (but same superficial symptom) will vote or comment on your ticket, thus making diagnosis of *your* issue harder.

When in doubt, log a new problem report

Many problems may not have any distinctive signature or steps to reproduce. In such cases, a new report is better than an upvote or a comment on another report, which is reporting a similar outward *symptom*. Depending on the type of report, include additional diagnostic files to your report as described later in this document.

Problem-specific best practices

Described below are problems that are hard to diagnose without good diagnostic files. After identifying the case that best describes your issue, follow the feedback steps specific to that case.

- **Crashes:** A crash occurs when the process (Visual Studio) terminates unexpectedly.
- **Unresponsiveness:** VS becomes unresponsive for an extended period of time.
- **Slowness issues:** Any specific action in VS is slower than desired
- **High CPU:** Extended periods of unexpectedly high CPU usage

Crashes

A crash occurs when the process (Visual Studio) terminates unexpectedly.

Directly reproducible crashes

Directly reproducible crashes are cases that have all the following characteristics:

- Can be observed by following a known set of steps

- Can be observed on multiple computers (if available)
- Can be reproduced in sample code or a project that can be linked to or provided as part of the feedback (if the steps involve opening a project or document)

For these issues, follow the steps in "[How to Report a Problem](#)" and be sure to include:

- The steps to reproduce the problem
- A standalone repro project as described above. If standalone repro is not possible, then please include:
 - The language of the open projects (C#, C++, etc.)
 - The kind of project (Console Application, ASP.NET, etc.)

NOTE

Most valuable feedback: For this case, the most valuable feedback is the set of steps to reproduce the issue along with sample source code.

Unknown crashes

If you're not sure what's causing your crashes or they seem random, then you can capture dumps locally each time Visual Studio crashes and attach those to separate feedback items. To save dumps locally when Visual Studio crashes, run the following commands in an administrator command window:

```
reg add "HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\Windows Error Reporting\LocalDumps\devenv.exe"

reg add "HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\Windows Error Reporting\LocalDumps\devenv.exe" /v DumpType /t REG_DWORD /d 2

reg add "HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\Windows Error Reporting\LocalDumps\devenv.exe" /v DumpCount /t REG_DWORD /d 2

reg add "HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\Windows Error Reporting\LocalDumps\devenv.exe" /v DumpFolder /t REG_SZ /d "C:\CrashDumps"
```

Customize the dump count and dump folder as appropriate. Find more information on these settings [here](#).

NOTE

Dumps captured using Task Manager are likely to be of the wrong bitness, which makes them less usable. The procedure described above is the preferred way for capturing a heap dump. If you do want to use Task Manager, close the one that is currently running, launch the 32bit Task Manager (%windir%\syswow64\taskmgr.exe) and collect a heap dump from there.

NOTE

Each dump file produced by this method will be up to 4 GB in size. Make sure to set DumpFolder to a location with adequate drive space or adjust the DumpCount appropriately.

Each time Visual Studio crashes, it will create a dump file **devenv.exe.[number].dmp** file in the configured location.

Then, use Visual Studio's "Report a Problem..." feature. It will allow you to attach the appropriate dump.

1. Locate the dump file for the crash you are reporting (look for a file with the correct Creation time)

2. If possible, zip the file (*.zip) to reduce its size before submitting feedback
3. Follow the steps in "[How to Report a Problem](#)", and attach the heap dump to a new feedback item.

NOTE

Most valuable feedback: For this case, the most valuable feedback is the heap dump captured at the time of the crash.

Unresponsiveness

VS becomes unresponsive for an extended period of time.

Directly reproducible Unresponsiveness

As described in the corresponding section on crashes, for problems that can be easily reproduced, seen on multiple machines and can be demonstrated in a small sample, the most valuable feedback reports are ones that include steps to reproduce the problem, and include sample source code that demonstrates the problem.

Unknown Unresponsiveness

If an unresponsiveness manifests itself in an unpredictable fashion, on the next occurrence, launch a new instance of Visual Studio and report a problem from that instance. In the "["Record" screen](#)", be sure to select the Visual Studio session that is unresponsive.

If the Visual Studio instance that is unresponsive was launched in Administrator mode, then the second instance would also need to be launched in Administrator mode.

NOTE

Most valuable feedback: For this case, the most valuable feedback is the heap dump captured at the time of the Unresponsiveness.

Slowness and High CPU Issues

What makes a slowness or high CPU usage issue most actionable is a performance trace captured while the slow operation or high CPU event is in progress.

NOTE

When possible, isolate each scenario in a separate, specific feedback report. For example, if typing and navigation are both slow, follow the steps below once per issue. This helps the product team isolate the cause of specific issues.

For best results in capturing the performance, follow these steps:

1. If not already running, have a copy of Visual Studio open where you will reproduce the problem
 - Have everything set up to reproduce the problem. For example, if you need a particular project to be loaded with a specific file opened, then be sure both of those steps are complete before proceeding.
 - If you are *not* reporting a problem specific to loading a solution, try to wait 5-10 minutes (or more, depending on solution size) after opening the solution before recording the performance trace. The solution load process produces a large amount of data, so waiting for a few minutes helps us focus on the specific problem you are reporting.
2. Start a second copy of Visual Studio *with no solution open*

3. In the new copy of Visual Studio, open the **Report a Problem** tool
4. Follow the steps in [How to Report a Problem](#) until you reach the "Provide a trace and heap dump (optional)" step.
5. Choose to record the first copy of Visual Studio (the one encountering performance problem) and start recording.
 - The Steps Recorder application will appear and begin recording.
 - **During the recording**, perform the problematic action in the first copy of Visual Studio. It is difficult for us to correct specific performance problems if they do not appear within the recorded time.
 - If the action is shorter than 30 seconds and can be easily repeated, repeat the action to further demonstrate the problem.
 - For most cases, a trace of 60 seconds is sufficient to demonstrate the problems, especially if the problematic action lasted (or was repeated) for more than 30 seconds. The duration can be adjusted as necessary to capture the behavior you would like fixed.
6. Click "Stop Record" in Steps Recorder as soon as the slow operation or high CPU event you want to report is finished. It may take a few minutes to process the performance trace.
7. Once complete, there will be several attachments to your feedback. Attach any additional files that may help reproduce the problem (a sample project, screenshots, videos, etc.).
8. Submit the feedback.

While recording a performance trace, if the slow operation or high CPU you are reporting comes to an end, then immediately stop the recording. If too much information is collected, the oldest information gets overwritten. If the tracing is not stopped soon (within a few seconds) after the interesting operation, useful trace data will get overwritten.

Do not directly attach performance traces to existing feedback items on Developer Community website. Requesting/providing additional information is a supported workflow in Visual Studio's built-in Report a Problem tool. If a performance trace is required in order to resolve a previous feedback item, we will set the state of the feedback item to "Need More Info", which can be responded to in the same way as reporting a new problem. For detailed instruction, please refer to "["Need More Info" section](#)" in Report a Problem tool's document.

NOTE

Most valuable feedback: For almost all slowness/high CPU issues, the most valuable feedback is a high-level description of what you were trying to do, along with the performance trace (*.etl.zip) which captures the behavior during that time.

Advanced Performance Traces

Trace collection capabilities in the Report-a-problem tool are sufficient for most scenarios. But there are times where more control over trace collection is needed (for example, trace with a larger buffer size), in which case PerfView is a great tool to use. Steps for manually recording performance trace using the PerfView tool can be found on the [Recording performance traces with PerfView](#) page.

See also

- [Visual Studio feedback options](#)
- [Report a problem with Visual Studio for Mac](#)
- [Report a problem with C++](#)
- [Visual Studio Developer Community](#)

- Developer Community data privacy

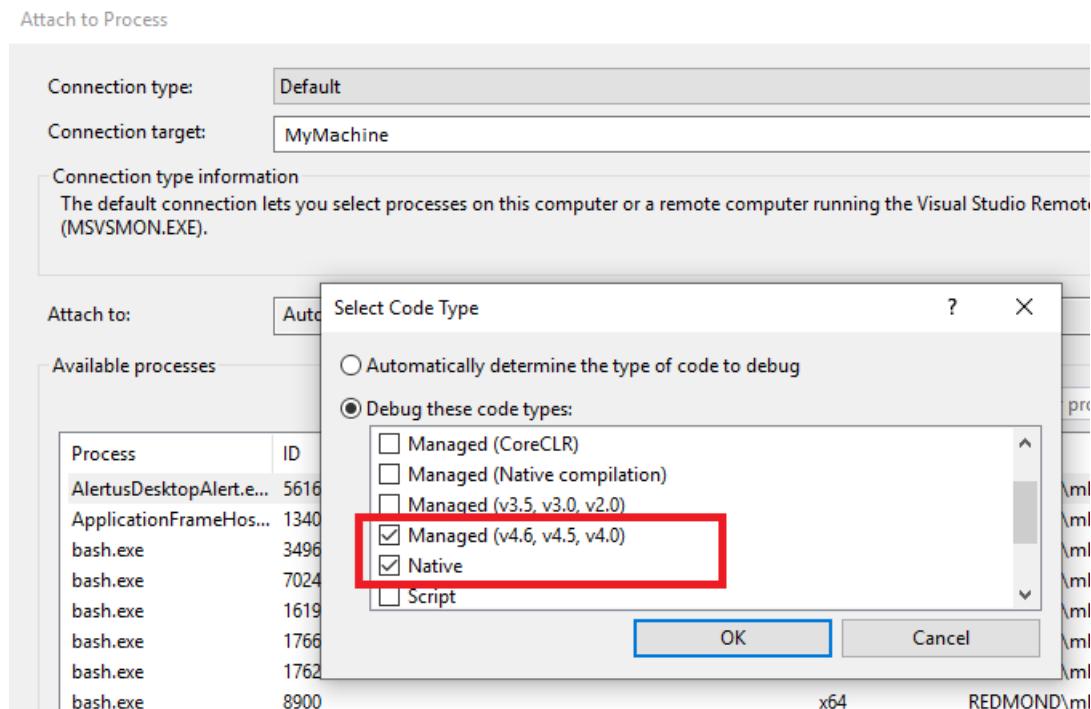
Create minidumps for a Visual Studio process with all call stacks

7/24/2019 • 2 minutes to read • [Edit Online](#)

In some cases, Microsoft might ask for a minidump of a running Visual Studio process with information for all call stacks. To collect this information, perform these steps:

Create the minidump file

1. Start a new instance of Visual Studio.
2. From the main menu, choose **Debug > Attach To Process**.
3. Check the relevant **Managed** and **Native** check boxes and press **Attach**.



4. Select the other Visual Studio instance to attach to from the list of running processes.
5. From the main menu, choose **Debug > Break All**.
6. From the main menu, choose **Debug > Save Dump As**.

Get the call stacks from the minidump

1. Open the dump file in Visual Studio.
2. Go to **Tools > Options > Debugging > Symbols** and make sure that **Microsoft Symbol Servers** is checked in the **Symbol file (.pdb) locations**.
3. Open the **Command window** (**View > Other Windows > Command Window**)
4. Type '`~*k`'. The window displays all threads' call stacks.
5. Copy all text from Command Window and save to a text file.
6. Attach the txt file to the bug.

Troubleshoot and create logs for MSBuild problems

7/24/2019 • 2 minutes to read • [Edit Online](#)

The following procedures can help you diagnose build problems in your Visual Studio project, and, if necessary, create a log to send to Microsoft for investigation.

A property value is ignored

If a project property appears to be set to particular value, but has no effect on build, follow these steps:

1. Open the Visual Studio Developer Command Prompt that corresponds to your version of Visual Studio.
2. Run the following command, after substituting the values for your solution path, configuration, and project name:

```
msbuild /p:SolutionDir="c:\MySolutionDir\";Configuration="MyConfiguration";Platform="Win32" /pp:out.xml  
MyProject.vcxproj
```

This command produces a "preprocessed" msbuild project file (out.xml). You can search that file for a specific property to see where it is defined.

The last definition of a property is what the build consumes. If property is set twice, the second value overwrites the first. Also, MSBuild evaluates the project in several passes:

- PropertyGroups and Imports
- ItemDefinitionGroups
- ItemGroups
- Targets

Therefore, given the following order:

```
<PropertyGroup>  
    <MyProperty>A</MyProperty>  
</PropertyGroup>  
<ItemGroup>  
    <MyItems Include="MyFile.txt"/>  
</ItemGroup>  
<ItemDefinitionGroup>  
    <MyItems>  
        <MyMetadata>$({MyProperty})</MyMetadata>  
    </MyItems>  
</ItemDefinitionGroup>  
<PropertyGroup>  
    <MyProperty>B</MyProperty>  
</PropertyGroup>
```

The value of "MyMetadata" for "MyFile.txt" item will be evaluated to "B" during build (not "A" and not empty)

Incremental build is building more than it should

If MSBuild is unnecessarily rebuilding a project or project item, create a detailed or binary build log. You can search the log for the file that was built or compiled unnecessarily. The output looks something like this:

```
Task "CL"

Using cached input dependency table built from:

F:\test\Project1\Project1\Debug\Project1.tlog\CL.read.1.tlog

Outputs for F:\TEST\PROJECT1\PROJECT1\PROJECT1.CPP:
F:\TEST\PROJECT1\PROJECT1\DEBUG\PROJECT1.OBJ
Project1.cpp will be compiled because F:\TEST\PROJECT1\PROJECT1\PROJECT1.H was modified at 6/5/2019 12:37:09 PM.

Outputs for F:\TEST\PROJECT1\PROJECT1\PROJECT1.CPP:
F:\TEST\PROJECT1\PROJECT1\DEBUG\PROJECT1.OBJ

Write Tracking Logs:
Debug\Project1.tlog\CL.write.1.tlog
```

If you are building in the Visual Studio IDE (with detailed output window verbosity), the **Output Window** displays the reason why each project is not up-to-date:

```
1>----- Up-To-Date check: Project: Project1, Configuration: Debug Win32 -----
1>Project is not up-to-date: build input 'f:\test\project1\project1\project1.h' was modified after the last
build finished.
```

Create a binary msbuild log

1. Open the Developer Command Prompt for your version of Visual Studio
2. From the command prompt, run one of the following commands. (Remember to use your actual project and configuration values.):

```
Msbuild /p:Configuration="MyConfiguration";Platform="x86" /bl MySolution.sln
```

or

```
Msbuild /p:/p:SolutionDir="c:\MySolutionDir\";Configuration="MyConfiguration";Platform="Win32" /bl
MyProject.vcxproj
```

A Msbuild.binlog file will be created in the directory that you ran MSBuild from. You can view and search it by using the [Msbuild Structured Log Viewer](#).

Create a detailed log

1. From the Visual Studio main menu, go to **Tools > Options > Projects and Solutions >Build and Run**.
2. Set **Msbuild project build verbosity** to **Detailed** in both combo boxes. The top one controls build verbosity in the **Output Window** and the second one controls build verbosity in the <projectname>.log file that is created in each project's Intermediate directory during build.
3. From a Visual Studio developer command prompt, enter one of these commands, substituting your actual path and configuration values:

```
Msbuild /p:Configuration="MyConfiguration";Platform="x86" /fl MySolution.sln
```

or

```
Msbuild /p:/p:SolutionDir="c:\MySolutionDir\";Configuration="MyConfiguration";Platform="Win32" /fl  
MyProject.vcxproj
```

An Msbuild.log file will be created in the directory that you ran msbuild from.

Collect an ETL trace with PerfView

10/25/2019 • 2 minutes to read • [Edit Online](#)

PerfView is a tool that creates ETL (event trace log) files based on [Event Tracing for Windows](#) that can be useful in troubleshooting some kinds of issues with Visual Studio. Occasionally when you report a problem, the product team may ask you to run PerfView to collect additional information.

Install PerfView

Download PerfView from [GitHub](#).

Run PerfView

1. Right-click on **PerfView.exe** in Windows Explorer and choose **Run as administrator** as admin
2. On the Collect menu, choose **Collect**
3. Check **Zip**, **Merge**, and **ThreadTime**.
4. Increase **Circular MB** to 1000.
5. Change **Current Dir** to save ETL traces to a specified folder and Data File if you are going to collect more than once.
6. To start recording data, choose the **Start Collection** button.
7. To stop recording data, choose the **Stop Collection** button. The PrefView.etl.zip file will be saved in the specified directory.

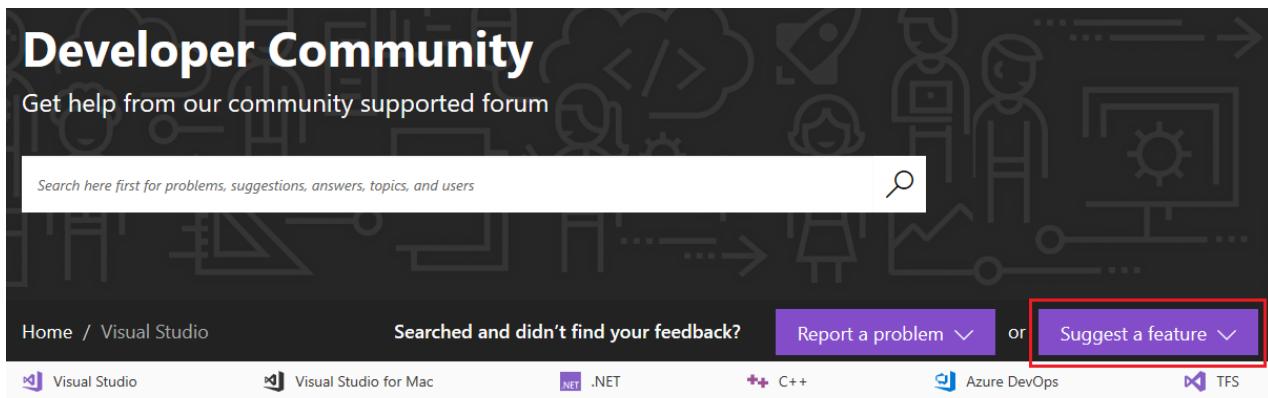
PerfView can store only the most recent data that fits into its buffer. Therefore, try to stop the collection as soon as possible after Visual Studio starts to freeze or slow down. Don't collect for more than 30 seconds after you hit a problem.

For more information, see [PerfView Tutorial on Channel9](#).

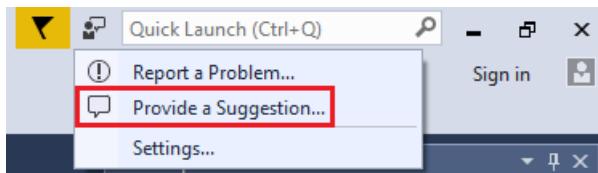
Suggest a feature for Visual Studio

12/5/2019 • 5 minutes to read • [Edit Online](#)

We've introduced a new experience for suggesting features alongside the current ability to report problems on [Visual Studio Developer Community](#). This is a new way you can be empowered to engage directly with Visual Studio's engineering workflow.



You can also start a feature suggestion directly from Visual Studio by choosing **Provide a Suggestion** from the **Provide Feedback** icon near the top right of the main Visual Studio window:



Choosing **Provide a Suggestion** takes you to [Developer Community](#), where you can enter your suggestion.

User Voice

Until now, Visual Studio users proposed new feature ideas on User Voice. By adding the *Suggest a feature* functionality to Developer Community, all of your feedback for the Visual Studio product team is now in one place.

The User Voice site has been made read-only. You can look back as you need to for context, but all new feature suggestions should be submitted on Developer Community.

We migrated an initial set of ideas from the User Voice forum to Developer Community. Migration was done based on the community impact of the feedback and our product roadmap priorities. If you were expecting to see a suggestion here that we may have missed, feel free to add it to Developer Community.

Votes

The voting system in Developer Community is different to User Voice. To maintain the integrity of Developer Community vote counts and to avoid skewing of votes, we show the User Voice vote counts prominently as a tag next to each feature suggestion. As you continue to vote on suggestions in the enhanced Developer Community, your new vote counts will show separately from the User Voice vote counts.

Suggestion status

After you submit a feature suggestion, states indicate where your feature submission is in its lifecycle. As we take

your feedback into consideration and move it along the workflow, we tag it with the corresponding state. The various states associated with feature suggestions are listed here, along with a description of their meaning and color indicators.

New

New means the suggestion has been newly reported from you or someone else. No action has been taken on it yet. The front line will do some preliminary checks to make sure we can proceed further. Expect to hear from us in about five business days with our next steps.

Under Review

Under Review indicates that the feature suggestion has been queued up for prioritization. We prioritize features to bring our broader developer community the best value, also taking the product roadmap into consideration.

Even if we're unable to pursue your new feature suggestion immediately, we'll continue to monitor your idea for about 90 days, let the community weigh in, and then make a decision on the next steps.

On Roadmap

On Roadmap means that your feature suggestion has a broad community impact and will help improve the product experience. We've allocated time for it on our roadmap. We'll update you on the progress.

Need More Info

A feature suggestion marked **Need More Info** means that we need more details so we can better understand your suggestion. Check the comments, where we'll ask for additional information to get a deeper understanding.

Closed - Other Product

Closed - Other Product means we're unable to address your feature suggestion at this time because it's not applicable to the product it was reported for. However, we'll provide details on where you can share your new feature suggestion for the appropriate product.

Closed - Duplicate

Closed - Duplicate indicates that someone else has already suggested the same feature. Review the comments to find the link to the existing feature suggestion. Votes and comments have been merged in the original suggestion. Follow the original suggestion.

Closed - Not Enough Info

Closed - Not Enough Info indicates that after several attempts, we haven't received enough information to understand your feature suggestion fully. We have to close the new feature suggestion as we're unable to take any further action at this stage.

Found the information we were looking for? You may request to reactivate the ticket when you have the additional information.

Completed - Preview

Completed - Preview indicates that we implemented the feature you suggested. You can download a preview

version of Visual Studio that contains your suggestion using the link provided in the comments.

Completed – Release

Completed - Release indicates that your new feature suggestion has been released in the latest product update. The Visual Studio update can be downloaded using the link provided in the comments.

FAQ

Why can't I see my User Voice idea in Developer Community?

New feature suggestions from the old User Voice forum have been migrated to Developer Community based on the impact to the broader community and our product road map priorities. If you think we've missed migrating your suggestion, add it as a new suggestion to Developer Community.

Why have the votes not been carried over from User Voice?

The voting system in User Voice operates differently from the voting system in Developer Community. In the new system, we want to maintain the integrity of the vote counts and avoid skewing the data. We decided to show the User Voice vote counts prominently as a tag (instead of votes) for each suggestion. When users vote on suggest a feature submissions using Developer Community, the new votes will show separately from the User Voice votes.

Where can I see comments associated with the suggestions imported from User Voice?

Follow the link we've included to go back to the User Voice comments on a suggestion that has been migrated to Developer Community. This link will be available during the transition period for easy reference if you need more context.

Why can I see three vote counts for a suggestion?

When a suggestion has been imported from User Voice, you'll see three separate vote counts. The two vote counts shown as tags are the vote counts that the idea received originally on User Voice. The third vote count displayed to the left of the suggestion lets you see how many votes the suggestion is getting from Developer Community.

The voting system in User Voice operates differently from the votes in Developer Community. To maintain integrity of Developer Community vote counts and avoid skewing the data, we decided to show the User Voice vote counts prominently as a tag (instead of votes) for each suggestion.

How long can I expect actions to take?

We're fully committed to listening to your feature suggestions and taking actions to provide valuable experiences to our customers. Actions on suggestions are prioritized based on impact to the broader community. Even if we can't respond personally to every suggestion, we'll make sure that your feedback gets to the right team and is evaluated carefully.

The response time depends on the status your feedback is in. Please review the explanation of statuses in this document to understand the response times.

See also

- [Introducing 'Suggest a Feature' in Developer Community \(Visual Studio blog\)](#)

Developer Community data privacy

10/18/2019 • 2 minutes to read • [Edit Online](#)

By default, all information in problem reports on [Developer Community](#), including any comments and replies, is publicly visible. This is beneficial because it allows the entire community to see the issues, solutions, and workarounds that other users have found. However, if you're concerned about the privacy of your data or identity, you have options.

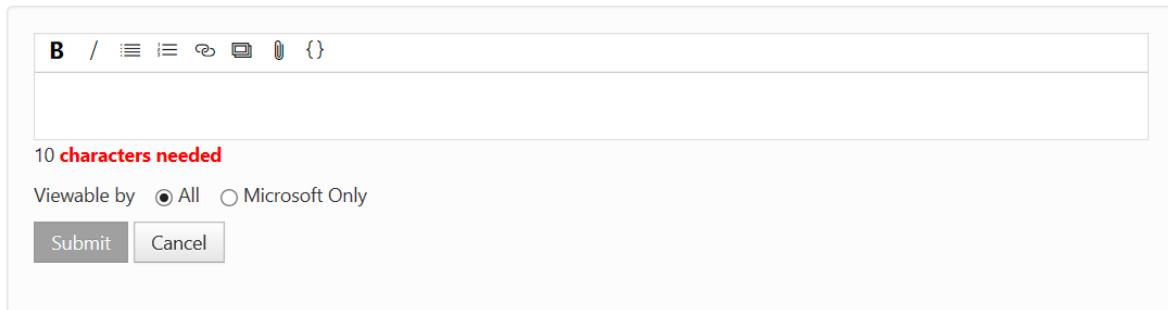
Identity privacy

If you're concerned about revealing your identity, [create a new Microsoft account](#) that does not disclose any details about you. Use this account to create your report.

Data privacy

If you're concerned about data privacy, don't put anything you want to keep private in the title or content of the initial report, which is always public. Instead, create the report, and then note that you'll send details privately in a separate comment. Once the problem report is created, you can specify who can see replies and attachments:

1. In the report you created, choose **Add comment** to create a private description of the problem.
2. In the reply editor, use the control below the **Submit** and **Cancel** buttons to specify the audience for your reply. Choose **Viewable by moderators and the original poster** to limit visibility to Microsoft employees and yourself.



Only the people you specify can see the comment and any images, links, or code you include in it. Any replies under the comment have the same visibility as the original comment. This is true even if the privacy control on replies doesn't show the restricted visibility status correctly.

3. Add the description and any other information, images, and file attachments needed for your repro. Choose the **Submit** button to send this information privately.

NOTE

There is a 2-GB limit on attached files, and a maximum of 10 files. If you need to upload a larger file, you can either submit a new problem report or request an upload URL from a Microsoft employee in a private comment.

To maintain your privacy and keep sensitive information out of public view, take care to keep all interactions with Microsoft to replies under a visibility-restricted comment. Replies to other comments may cause you to accidentally disclose sensitive information.

Data we collect

If **Report a problem** is initiated from Visual Studio Installer, we collect the most recent setup log.

If **Report a problem** is initiated from Visual Studio, we collect one or more of the following types of data:

- Watson and .NET entries from the event log
- Visual Studio in-memory activity log file
- PerfWatson files, if Watson collection is enabled
- LiveShare log files, if they exist
- Xamarin log files, if they exist
- Nuget log files, if they exist
- Web debugger log files, if they exist
- Service Hub logs and MEF error logs, if they exist
- Python logs, if they exist
- Windows Forms logs, if they exist
- A screenshot, if you choose to include it
- Recording data, if you choose to include a recording, which includes:
 - Steps to reproduce the problem
 - ETL trace file
 - Dump file

NOTE

Log files, screenshots and recording data are sent to Microsoft only when you provide permission by submitting the problem report with which they are included. You can see which files are included on the 'Summary' step of the 'Report a Problem' window (see the screenshot included in this note). Collected logs and files are stored in the %temp% folder and are cleaned up regularly and after each upload. If you don't want to include a log in your problem report, delete the file from the %temp% folder before submitting the report.

The screenshot shows the 'Report a Problem' window with the 'Summary' tab selected. It includes fields for 'My problem title', 'Steps to reproduce my problem', and sections for 'Screenshot/attachments' and 'System Logs'. A note at the bottom states that attachments will be private to Microsoft while user info and description will be public on the Developer Community.

Visual Studio Feedback

Report a Problem

Describe Attachments Summary

My problem title

Steps to reproduce my problem

Screenshot/attachments

Screenshot

System Logs

201909031956_D15.9_15.9.28307.812_72508_364feb75-fa67-4e05-8787-
201909032059_MASTER_16.3.29221.178_672_3dace10d-da93-4b39-bc93
devenv.isolation.ini
VSInMemoryActivityLog_2019-09-05_20-57-19.xml

Your attachments will be private to Microsoft. Your user name, title, and description will be public on [Developer Community](#).
For more on privacy, see the [Microsoft Privacy Statement](#).

Previous Submit

See also

- [How to report a problem with Visual Studio](#)
- [C++ problem report data privacy](#)