Due to the complexity of Visual Studio and the variety of projects you can write in it, simple screenshots or text descriptions are often not enough for us (the developers who work on Visual Studio) to be able to address a problem. We need to be able to reliably reproduce a problem first in order to understand what the issue is, and second to ensure that the fixes we provide fully address the problem.

This document is a guide to how to provide repro steps when you encounter a problem in Visual Studio. We provide here many different options, listed in *preference* order - please try to provide the first listed option that you are able to.

In addition to the data provided below, we always want to know the exact version of Visual Studio (including updates) that you have installed, what OS you're using, and any other configuration data that might be relevant to the problem at hand. Much of this information is provided during the Feedback Upload process, but listing it explicitly can also help other users confirm the issue and is always a good practice.

Steps in a New Project

The very best thing we can work with is also the simplest: If you can describe how to reproduce a problem by simply starting from one of the built-in project templates, that is often ideal.

Many project templates have similar names - be *extremely* specific! Saying "new web app" might refer to any of several dozen different templates; provide their *full names* with descriptions - "ASP.net Core Web Application, then choose the Blank template on the following dialog".

If you're using a template you installed from the web or gallery, we would prefer that you provide a zip file (see below). Matching the exact version of a template you may have downloaded months ago is very difficult and may not produce the same behavior.

A Zip File

If you're not sure how to reproduce the problem from a fresh project, a zip file containing your entire solution folder is often the next best thing. A Microsoft employee will provide you an email address to send it to, and we'll work with you 1:1 to determine what the issue is.

Sidebar: NDAs and Confidentiality Keeping the security and confidentiality of your code is absolutely critical to us. You can be assured that the contents and nature of any files you send will *not* be disclosed in any way. We will delete these files as soon as possible, and our interaction with your files will be kept to the minimum required to diagnose the underlying issue.

If your organization requires the signing of a Non-disclosure Agreement (NDA), in most cases we will be able to sign such an NDA to facilitate debugging an issue. Please follow up via email with the provided engineer and they will be able to provide more details.

Anonymization / Simplification If the content and names of your files are irrelevant to the issue at hand, a simplified/cleansed version of the above zip file is also very much acceptable. The smallest and simplest zip file that can reproduce the problem is always preferred - if you *can* send us a single project out of your solution that still reliably demonstrates the issue, we'd prefer it over a larger one.

A File Listing

If you're unable to provide a zip file, a file listing can be useful. This **PowerShell** command will produce a `files.csv` file listing the file names and lengths under a folder (replace `C:\MY_PROJECT` as needed):

```
Get-ChildItem C:\MY_PROJECT -Recurse -Depth 5 | Where
{!$_.PSIsContainer -or !$_.Attributes -match "ReparsePoint"}
| Select-Object FullName, Length | Export-Csv -noTypeInformation
-path files.csv
```

As above, you can post this in the issue comments, or send us the information in an email.

A file listing is usually not enough information without a TypeScript Server log (see below).

TypeScript Server Log

For issues involving TypeScript or JavaScript language service issues, a TypeScript Server log is extremely useful in diagnosing issues.

Complete documentation for setting this up can be found in the wiki. Here's a summary:

- Close Visual Studio
- From a command prompt:
  - Run `setx TSS_LOG "-level verbose -file C:\tmp\tsserver.log"` (you can use a different path than `C:\tmp`, naturally)
  - `mkdir C:\tmp` (TS Server won't create this by default)
- Start Visual Studio
- Reproduce the issue
- Close Visual Studio
- `setx TSS_LOG ""`
- Copy `C:\tmp\tsserver.log` as the next TypeScript/JavaScript language service will overwrite it

Some notes and tips about `tsserver.log`: * To minimize the log file size, open as few documents as needed. * Some file contents may be visible in the log. Treat this with the same security you would the code itself. * The log gets quickly gets very large. Perform the minimum steps to reproduce the issue and immediately close Visual Studio.