Terminal Issues

This document is about reporting issues for the integrated terminal (`ctrl+``). Learn more about how to use the terminal in the documentation.

Creating great terminal issues

- Make sure you read the common questions and long-standing known issues sections below as you might be reporting an issue that is already known.
- Include the VS Code version, Operating System version and a list of extensions you're using. Ideally you should use the issue reporter built into VS Code for this as it automatically includes this information in the report.
- Including a screenshot or gif is normally a good idea.
- Including your settings.json is also a good idea as many issues are normally related to bad configuration.
- Take note if you're using an extension to launch the terminal, for example using a debugger to launch the terminal, language extension (eg. PowerShell Integrated Console) or something like Code Runner. If so there's a good chance it's an issue with the extension, not with VS Code.

Windows-specific additions:

- Be sure to include the build number if you're on Windows 10, this is attached automatically when using the builtin issue reporter or can be fetched manually by running `ver` in `cmd.exe`.
- If you're on Windows 10 1809 and below then you will be on the old terminal backend called `winpty`, a lot of these issues will not be actionable and closed as the solution to them was to move to the new frontend `conpty`.

Common questions

Make sure you read over the common questions section on the website.

Long-standing known issues

Here are some long standing known issues in the terminal:

- Various emulation issues on Windows #45693
- Terminal does not show any text on Windows 7 #43169
  - Try turning off compatibility mode
- Characters like underscore are being cut off #35901
  - Try changing the `terminal.integrated.fontFamily`
- Emojis are printed as double width but are single width xtermjs/xterm.js#1059
- Non-English characters duplicated on Windows #132715

Which issues go in which repos

The terminal has several dependencies which are also open source projects such as xterm.js, node-pty and conpty. Managing issues is difficult across so many repos so the general rule we follow with terminal issues is that fairly niche upstream issues are only tracked in the upstream repositories and major upstream issues are tracked in VS Code as well in order to improve discoverability of the issue in question.

Diagnosing terminal issues

Enabling trace logging

For some terminal issues it's useful to get trace logs, this can reveal at what point something is failing. Follow these steps to get the logs:

1. Close all VS Code windows
2. Launch VS Code from the terminal using `code --log trace`
3. At this point you should reproduce the terminal issue you're having
4. Run the command "Developer: Open Log File..." (`F1` opened command palette) and select `Window` to get an editor containing the logs. Select the `Remote Extension Host` log file if you're diagnosing an issue in a remote workspace (SSH/WSL/Containers/Codespace).

If for some reason you're unable to restart VS Code like you're running in a remote, you can change the log level via the command palette (`F1` `Developer: Set Log Level...`).

Enabling escape sequence logging

For issues where text is misbehaving in the terminal you can enable logging of the data being sent to/from the terminal emulator and the shell process. To enable escape sequence logging run the "Terminal: Toggle Escape Sequence Logging" command from the command palette (`F1`), the logs can then be viewed in the devtools console (Help > Toggle Developer Tools).

Using showkey to investigate keybinding issues

There is a utility called `showkey` which will print the character codes as received by the application, this is similar to escape sequence logging above but's evaluated on the process side. Install `showkey` by installing the `kbd` package on Linux or `showkey` on homebrew, for example:

```
sudo apt update
sudo apt install kbd
showkey -a
```

```
brew install showkey
showkey -a
```

Rendering problems

Figuring out what's going on with rendering can be tricky as there are a lot of moving parts. A blank screen could mean that the terminal was never created properly and the terminal is fine, or maybe that the renderer is broken. Here are good steps to help find the root cause of rendering problems:

- Zoom in and out (ctrl/cmd++, ctrl/cmd+-) will force the renderer to redraw everything
- The terminal features webgl, 2d canvas and dom-based renderers. Changing the renderer type can identify issues with a particular renderer, you can turn off the canvas renderers with this setting:

  ```
  "terminal.integrated.gpuAcceleration": "off"
  ```

Known rendering problems:

- Corrupt texture showing after resuming OS from sleep #69665
- Underscore and similar chars not showing up #35901
- Characters become small or large after changing monitor DPI xtermjs/xterm.js#2137

Text wrapping problems

Sometimes terminal wrapping doesn't work as expected, for example a line that is expected to wrap will start overwriting the first part of the wrapped line. This type of problem is typically related to the backend and frontend's column count differing. You can get the frontend's column count by counting the number of cells in a line, for the backend `stty -a | grep columns` should tell you.

Why did you close my issue?

We get *a lot* of issues and have to split our time between responding to and issues and actually improving the product. Because of this, we have a pretty high bar on issue quality and reproducibility, if we can't reproduce the issue we may have to close it off since we cannot action it.

Similarly, we depend on several upstream components and may close an issue off if the problem is likely related to one of those. The primary example of this is "conpty" which is a dependency built by the Windows Terminal team and shipped as part of Windows. This is a much better situation than earlier as there is a team of experts focusing on Windows Terminal and its backend conpty, but they also bundle the latest version of conpty which we may get a year later when a Windows update ships. So we may close your issue as a conpty issue, even though it works fine in Windows Terminal because it's most likely fixed in a later version of Windows. Another side effect of this is our old backend "winpty" is now deprecated and we don't plan on improving it as the "fix" for problems in winpty is to move to the maintained official Microsoft backend conpty.

Whether we can action the issue is the main reason we close issues off but we may also close an issue as designed or out of scope. You can visit issue grooming for more info on how we manage and triage issues.