WinLine Documentation

This document contains all the information you need to get started using WinLine, including a brief overview of commands.

Document version: **1.0**

The information this document contains applies to WinLine **3.2** to WinLine **3.6**

Contents

[What is WinLine? 2](#_Toc119079647)

[Minimum Requirements 2](#_Toc119079648)

[Table Color Key 3](#_Toc119079649)

[Command Overview 3](#_Toc119079650)

[Components 5](#_Toc119079651)

[Management 5](#_Toc119079652)

[Component Files & Packs 5](#_Toc119079653)

[Component Pack Development 6](#_Toc119079654)

[Configuration File 6](#_Toc119079655)

[WinLine File Types 7](#_Toc119079656)

[Developer Information 7](#_Toc119079657)

[Error Codes 7](#_Toc119079658)

[Causes 7](#_Toc119079659)

[Solutions 8](#_Toc119079660)

[Component Packs – Manifest Format 8](#_Toc119079661)

[Component Packs – Folder Structure 8](#_Toc119079662)

# What is WinLine?

WinLine is a simple, but powerful, terminal emulator. It behaves like the GNOME terminal, with many of the same commands. WinLine is designed to provide a Linux-like experience on Windows, and to improve the lackluster Command Prompt.

WinLine is written in Python, a high-level language that inexperienced users can read (with some practice!). We chose to use Python to allow novice users to try their hand at modding, adding new features, and to make our code more accessible.

# Minimum Requirements

WinLine is an extremely lightweight tool and should be supported by most systems. If you have concerns about compatibility, please refer to the information below

(Officially) Supported Operating Systems:

* Windows 7 / 10 / 11\*
* Ubuntu 20.04
* Debian 10 or greater

\* Windows 7 and 11 have not been tested

(Probably) Supported Operating Systems (UNTESTED!):

* Windows 8 / 8.1
* Unix
* Fedora / RedHat
* Arch Linux
* Manjaro
* ElementaryOS
* POP!\_OS

Other operating systems may have support, but we could not reasonably test all operating systems

UNSUPPORTED Operating Systems:

* Windows Vista (or earlier)
* MacOS / iOS / iPad OS
* Android / Chrome OS\* / Fuchsia / WearOS

\* WinLine is probably supported on Chrome OS through Penguin

Recommended Specs:

* x86 or 64-bit CPU (either Intel or AMD)
* 4 GB RAM
* ~5.3 GB free space (to install Python, packages, WinLine, and components)
* SSD (an HDD can work but it will be significantly slower)

Software Requirements:

* Python 3.8 / 3.10
  + Tkinter (for GUI commands. WinLine will run without it, but some features may break)
  + Any packages required by components

Users are welcome (and encouraged!) to test WinLine on other operating systems and to post their findings on the GitHub page. This would be hugely helpful!

# Table Color Key

Only available on Windows

Similar functions, but cannot be used interchangeably

May have security vulnerabilities

**Confirmed to have serious security implications or unexpected side effects**

**Can have unexpected side effects**

Will be deprecated soon

# Command Overview

This section provides a brief overview of the commands provided by WinLine. The commands are listed in the order they were added to WinLine

[Brackets] indicate an OPTIONAL parameter, while <Chevrons> indicate a REQUIRED parameter

|  |  |  |
| --- | --- | --- |
| Command | Description | Syntax |
| help | List all commands | help |
| exit | Exit WinLine | exit |
| clear | Clear the terminal window | clear |
| cd | Move into a directory | cd <path> |
| ls | List contents of a directory | ls [path] |
| term | Create a new instance of WinLine | term –[flags] |
| del | Delete a file | del <path:file> |
| rmdir | Delete a folder | rmdir <path:directory> |
| cat | Print the contents of a file | cat <path:file> |
| open | Opens a file using the default program | open <path> |
| man | Get advanced help for a command | man <command> |
| ipaddrs | Get the device’s current IP address | ipaddrs |
| ping | Pings an IP address / hostname | ping <destination> [count] |
| top | List all running commands | top |
| kill | Kill a running / suspended process | kill <process> |
| clock | Launch a simple inline clock | clock |
| list-drives | List all connected storage devices | list-drives |
| monitor | Launch an inline system monitor tool | monitor |
| components | List, manage, install, and remove components | See COMPONENTS section for more information |
| config | Launch a file-select prompt to open the config file in your preferred text editor | config |
| change-name | Change the WinLine username | change-name <name> |
| user | Echo the current user’s name | user |
| battery-report | Generate a battery report (laptops only) | battery-report |
| cmd | Start cmd.exe inline | cmd |
| powershell | Start Powershell inline | powershell |
| **sysRun** | **Ignore built-in commands and components; run the command directly on the system** | **sysRun <command>** |
| mount\_folder | Mount a folder as a network drive | mount\_folder <net drive> <folder’s drive> <folder path> |
| unmount\_folder | Unmount a network drive | unmount\_folder <net drive> |
| wldata | Open the data folder | wldata |
| edition | Get the current terminal’s edition | edition |
| path | Print the current working directory | path |
| reconfigure | Update configuration to latest version | reconfigure |
| uninstall | Uninstall WinLine | uninstall |
| recovery | Recover data from a .wlc file | recovery |
| backup | Compress all user data (including components) into a .wlc file | backup |
| update | Attempt to update WInLine | update |

For more help with mount\_folder and unmount\_folder, run these commands without any parameters

When possible, **avoid using sysRun**. This command might result in system damage if you’re careless. **This command will not be available unless you enable advanced mode**

# Components

Components provide a convenient way to add custom commands and scripts to WinLine. Several components are shipped with WinLine by default, but you can install your own components using the management commands. **Please be aware that you should not install components from untrustworthy or unknown sources, as they can be used for malicious activity**.

Please be aware that components are currently only supported on Windows systems, but we are working on expanding this functionality to Linux.

## Management

There are multiple subcommands to manage components called “flags”. Please see the table below for more info

|  |  |  |
| --- | --- | --- |
| Flag | Description | Example |
| --unload | Unloads a component, preventing it from running | components --unload <component> |
| --load | Forcefully load a component, even if it otherwise would be unloaded | components --load <component> |
| --disable | Disables a component | components --disable <component> |
| --enable | Forcefully enable a component, even if it otherwise would be disabled | components --enable <component> |
| --purge | Permanently uninstall a component | components --purge <component> |
| --install | Install a component or component pack | components --install |

## Component Files & Packs

There are two separate ways to share a component: a .py file (single component) or a .wcp (Component Pack) file.

A single component should contain dependency imports, internal functions, and **all code** within the .py file. It should NOT access external resources. This is a good entry-level interface to modding WinLine.

However, component packs can include multiple files, resources, and more! Component packs provide a huge benefit over single components as they can be much more complex, spanning as many files as you desire. *You could even ship an entire game in a component pack!*

## Component Pack Development

Component packs need to have a very specific folder structure and must be packaged in a very specific way for them to work correctly. Please refer to the information below to learn more.

**Note**: If you find this section too confusing, you can always download the example\_package.wcp file from our GitHub page.

**Basic Folder Structure** (once PACKAGED)

**package\_name.wcp**

* **manifest.json**
* **main.py**
* **package\_name.py**

Where manifest.json contains information about the package, main.py contains your MAIN code (what you would normally execute to run the command), and package\_name.py contains the code linking the package to WinLine. You can obtain examples of these files from our GitHub page.

# Configuration File

The configuration file controls all of WinLine’s preferences. It can be accessed using the **config** command, which will open a Choose Application dialogue. You can open it with any text editor.

Here’s what the configuration file contains, what the settings do, accepted values, and default values

|  |  |  |  |
| --- | --- | --- | --- |
| Setting Name | Description | Accepted Values | Default |
| keyboardInturruptEnabled | Weather to exit on Ctrl+C or not | true/false | true |
| cmdLink | Deprecated |  |  |
| advancedMode | Weather to enable dangerous commands | enabled/disabled | disabled |
| sysrun\_failed\_commands | Automatically run a command through the system if it isn’t built-in | true/false | false |
| allow\_components | Weather to enable non-developer components or not | true/false | true |
| emulate\_linux | (Windows only) Weather to emulate a Linux instance or not | true/false | false |
| display\_username\_win | (Windows only) Weather to show the user’s name as a location prefix | true/false | true |
| prefer\_reset\_on\_error | Controls weather WinLine resets to a blank console on error  true: Reset to blank console  false: warn of a fatal error but continue in current instance | true/false | false |
| enable\_malware\_protection | Weather to check for malicious components on startup | true/false | true |
| threaded\_mw\_protection | Weather to check for malicious components in the main thread or a secondary thread. Enabling this hides detection warnings. No effect if malware protection is disabled | true/false | true |
| safe\_mode | Weather to disable components (regardless of allow\_components) and disable dangerous commands | true/false | false |

# WinLine File Types

WinLine has multiple custom file types. Here’s a list of them:

|  |  |  |
| --- | --- | --- |
| File Extension | Type | Description |
| .wcp | WinLine Component Package | A package containing an advanced component or a set of components |
| .wlc | WinLine Backup File | Contains recovery data for use with the **recovery** command |

# Developer Information

## Error Codes

This section lists all error codes, their cause, and the solution

### Causes

|  |  |
| --- | --- |
| Error Code | Cause |
| 716a5e08 | The component package is using a manifest version not supported by the current version of WinLine. This might be solved by updating |
| 78bfc3343513c0 | The component package is intended for an older version of WinLine. The current version of WinLine does not support this component anymore |
| 35dc7e68 | The component package has an invalid manifest format, or the manifest version is incorrect (not a float) |
| a1797e78 | WinLine can’t reach our servers for online functions (like downloading updates) |
| e0b7e0dc5ce0 | Relative navigation couldn’t complete the requested operation. |
| 78be2e70fc | A dependency for the command failed to install |
| 50bc2e19a0 | Generic error message |
| d07c670a95 | The component package does not have a manifest |

### Solutions

|  |  |
| --- | --- |
| Error Code | Solution |
| 716a5e08 | Run the “update” command to update WinLine |
| 78bfc3343513c0 | Open the [GitHub](https://github.com/Psychon-Dev-Studios/new_winline/releases) page and download an older version of WinLine.  THIS WILL OVERRIDE YOUR CURRENT INSTALLATION |
| 35dc7e68 | Report this issue to the developer on the [discussions page](https://github.com/Psychon-Dev-Studios/new_winline/discussions/7), and wait for them to fix the issue |
| a1797e78 | Connect to the internet and verify your device can reach <https://psychon-dev-studios.github.io> |
| e0b7e0dc5ce0 | Check that the folder that you’re navigating into and folder you’re navigating from still exist |
| 78be2e70fc | Connect to the internet, verify your device can connect to pypi.org, and try again. If this doesn’t work, use pip to install tkinter |
| d07c670a95 | Report this issue to the developer on the [discussions page](https://github.com/Psychon-Dev-Studios/new_winline/discussions/7), and wait for them to fix the issue |

## Warning Codes

|  |  |
| --- | --- |
| Warning Code | Definition |
| 605b92fde67380 | The backup was created on a different version of WinLine. Restoring this backup might break some features or config options |
| 7b4fafd0 | Built-in A/V detected malicious components and has quarantined them |
| f0742100 | An internal variable caused a command to fail |
| 3d05ce00 | The operating system returned an unhandled error |
| 3dc5c76cf6 | An overflow error occurred |
| e0b9c3f0 | Maximum recursion depth reached |
| a1282e70 | An internal Python error occurred. The problem was severe enough to cause an exception, but not enough to cause a Python error |

## Component Packs – Manifest Format

This is the manifest format with default values. This is the format for Manifest v**1.1**

{

    "package\_name": "example",

    "package\_version": "1.0.0",

    "file\_onExit": "/myfile.txt",

    "format\_version": 1.1

}

If the component does NOT need to open a file when installed, set “file\_onExit” to “none”, like so:

"file\_onExit": “none"

Replace “file\_onExit” with the RELATIVE PATH to the file that should be opened when installed. For example: “/subfolder/myfile.txt”

## Component Packs – Folder Structure

pack\_name.wcp

/component\_name.py

/main.py

(whatever else you need)

This is the folder structure for a component pack. This is only the bare minimum structure,

You can obtain the template for “component\_name.py” from our [Official Components page](https://github.com/Psychon-Dev-Studios/new_winline/discussions/7). Remember to replace the template data with your own data.

Component\_name.py should be renamed to your component name, and main.py should contain your actual component code