



Under construction.

Windows (WINE)

WINE will play Windows games, both old and recent games, both 32-bit and 64-bit. Some open-source games are available in the free Batocera content store (from the EmulationStation menu, **UPDATES & DOWNLOADS → CONTENT DOWNLOADER**)



The following links contain info about some games' compatibility with WINE:

- [List liberodark](#)
- The [Windows-games-on-Batocera spreadsheet](#). [Direct link to Google sheets document](#).
- [WineHQ AppDB](#)
- [ProtonDB](#)
- [supertuxland](#)



It is recommended to use **btrfs** or **ext4** for /userdata/ as WINE [explains on their FAQ](#) that some games won't run on a non-Linux native file system. Steam games are [notoriously known to be unusable under WINE when using the NTFS filesystem](#).



In Batocera **v32** and higher there is a bug where WINE cannot run applications/games stored on a NAS. This can be worked around by not using a NAS for the saves folder.

This system scrapes metadata for the "pc" group and loads the windows set from the currently selected theme, if available.

Quick reference

- **Emulator:** [wine](#)
- **Cores available:** [wine: lutris](#), [wine: proton](#)
- **Folder:** /userdata/roms/windows/

- **Accepted ROM formats:** .pc, .exe, .wine, .wsquashfs, .wtgz

BIOS

No Windows emulator in Batocera needs a BIOS file to run.

Program files

- /userdata/roms/windows/: installed windows game (both 32 and 64-bit).
- /userdata/roms/windows_installers/: .iso, .msi or .exe used to install games into /userdata/roms/windows/.

Games coming with an installer

There are 2 types of formats for a Windows game installer:

- CD-ROM images: An *.iso file created from a game on a CD-ROM
- Executable Installers: A *.exe or *.msi file, that usually contains “setup” or “install” in the name (in some cases, other files may be bundled with them)

In both cases, those files need to be put into the /userdata/roms/windows_installers/ folder. Once this is done, refresh the gamelist in EmulationStation, then go into the “Windows” system, select “Install a new game” or select “Windows Install”-system, and run the installer file.

To make the “Install a new game” visible within the “Windows” system, go to **Main Menu -> Game Collection -> Disable “Don't show groups with only one system”** (This is imho the better option - Windows Installer isn't a real system, even it can be challenging to install some Windows-based games - thank you Mr. Gates)

After a screen announcing that the configuration is being made, the installer should open up. It'll look the same way it does inside Windows.



Some installers themselves may require additional dependencies to run. If you're not sure what they are, you could consider running the installer on a real Windows machine and then copying its installed data from that machine directly into /userdata/roms/windows/.

Proceed to install the game in the normal way, as for the installation path, leave it to the default: The virtual C:\ Drive will be located at /userdata/roms/windows/<game_name>.wine/drive_c/.

Once the installation is done, it tries to automatically create the autorun.cmd file based on the executable files present. If there are several exes or you need to set up command parameters, then head over to the /userdata/roms/windows/<game_name>.wine/ folder, and edit the autorun.cmd accordingly.



The file must use Linux line terminators! Do not use Notepad or other similar Windows-oriented text editors; use Notepad++ or Atom and select Unix (LF) as its line termination format.

Example:

```
DIR=drive_c/Program Files/xmoto 0.6.1  
CMD=xmoto.exe --fullscreen
```

or if there are some spaces in the name of the game:

```
DIR=drive_c/Program Files (x86)/Ubisoft/Rayman Origins  
CMD="Rayman Origins.exe"
```

This <game name>.wine folder is called the wineprefix.

In Batocera **42 and higher** it is possible to add a save directory argument to specify where that game saves files to in the Wine prefix. These directories will be transferred to /userdata/saves/windows/<GAME> and are symlinked to the game specific wineprefix. So in case you have to rebuild all wineprefixes (version change for example), your savegames are still available and working.

Example for a directory:

```
SAVEDIR=drive_c/users/root/documents/localappdata/game/  
DIR=drive_c/game  
CMD=game.exe
```

Example for files (Use ; as delimiter):

```
SAVEDIR=drive_c/game  
SAVEFILES=savefile1.sav;savefile2.save  
DIR=drive_c/game  
CMD=game.exe
```

There are some additional parameters allowed, ENV and LANG. The language can be changed per game for WINE. It is needed for certain older Windows Unicode games (mostly Japanese). If not used, games with Unicode filenames may fail to launch, and other games may display text incorrectly. May require additional fonts to be installed via winetricks, depending on the game. Use LANG=[language code]. The ENV parameter allows the adding of any other parameter, it is quite useful for [overriding DLL files](#) but read more about in it's chapter section.

```
ENV=MESA_EXTENSION_MAX_YEAR=2002  
LANG=ja_JP.UTF-8
```



If the path isn't working, try putting ./ before the path. Eg. DIR=./drive_c/Program



Files/xmoto 0.6.1

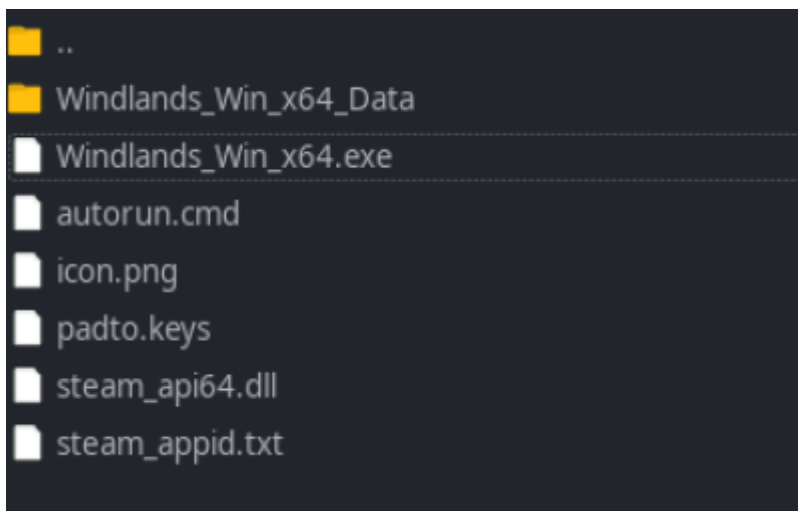
Games that don't need any installation

Those games usually come as a game folder containing a *.exe file (or several) as well as other files and subfolders. They may also come as a *.zip file, if that is the case, unzip them first.

For those, there are two easy ways to handle them:

- The EXE method: Just put the game folder into the /userdata/roms/windows, and refresh the gamelist in EmulationStation
- The PC folder method (recommended): Rename the game folder as <game name>.pc (eg. Windlands.pc), put it into the /userdata/roms/windows/, create a file named autorun.cmd directly into it, edit it so it points to the game executable.

Example:



If the *.exe of the game is directly in the <game name>.pc/ folder, and is called Windlands_Win_x64.exe, the autorun.cmd file should contain:

```
CMD=Windlands_Win_x64.exe
```

If the name of the executable contains special characters or spaces however, be sure to put quotes around it, like so:

```
CMD="launch game.exe"
```

If the EXE is instead inside of a subfolder, the DIR command needs to be added first. For example, if the game is at Game name.pc/64bit/bin/folder with space in its name/Game name 64-bit.exe and requires special launch parameters:

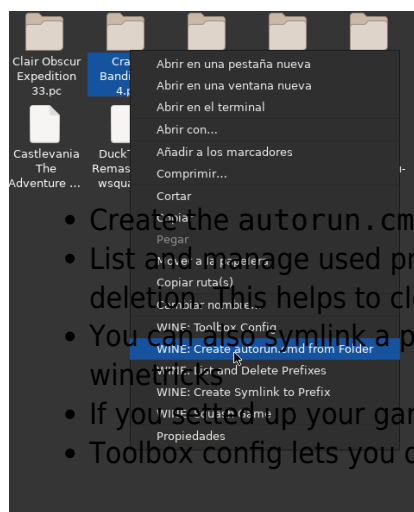
```
DIR=64bit/bin/folder with space in its name  
CMD="Game name 64-bit.exe" --fullscreen
```

The first time a game is run, a <game name>.wine folder will be created at /userdata/saves/windows/proton/ or /userdata/saves/windows/.

This <game name>.wine folder is called the wineprefix.

Use the WINE.toolbox for setup (v43) and a clean structures

Please read and understand the text written above as this reflects the basics how to setup WINE games. Since **v43** the process of creating the autorun.cmd, managing prefixes, compressing games, extraction of games is made with more convenience.



Press F1 to enter [PCManFM](#) and navigate to /userdata/roms/windows. By making a left mouse click on your windows game folders you can see several commands that will help you to setup and manage all your windows games.

- Create the autorun.cmd by selecting a single exe out of list
- List all the usage used prefixes for all games, prefixes not bound to a prefix will be selected for deletion. This helps to clean up your prefixes and saves also a lot of space
- You can also symlink a prefix to a default one, this helps if you setted up a working prefix with wine
- If you setted up your game it is a good idea to compress/squash it
- Toolbox config lets you open a dialog to set text size

Some more tricks:

- Use symlinks for custom runners, so you have installed wine-11.0-amd64 into /userdata/system/wine/custom/wine-11.0-amd64 then you symlink this to /userdata/system/wine/custom/gog-wine-11.0-amd64 this keeps wine prefixes separated.
- winetricks can carry more than one winetrick now, usage batocera-wine windows tricks <game.pc/wine> trick1 trick2 ... trick{n}
- Some winetricks need a pure 32bit prefix. All prefixes will be installed as 64bit variant. So tick Enable 32bit within ES Advanced Game Options (This has to be done **before** the prefix is created! If the prefix was created, delete it with the file explorer and start over again
- If you symlink a wine runner you can use win32-<wine runner> the word win32- will enable the build of 32bit prefixes if this runner is selected. This also helps to keep things tidy and separated.

Emulators

WINE

WINE configuration

Standardized features available to all cores of this emulator: windows.videomode,

windows.padtokeyboard

ES setting name batocera.conf_key	Description ⇒ ES option key_value
DXVK windows.dxvk	Converts the DirectX 9/10/11/12 calls to Vulkan. Should only be used on hardware that natively supports Vulkan. ⇒ Off 0, On 1.
DXVK HUD windows.dxvk_hud	See your FPS and version of Vulkan API/driver. ⇒ Off 0, On 1.
ESYNC windows.esync	Removes wineserver overhead for synchronization objects. Can increase performance for games that stress the CPU. ⇒ Off 0, On 1.
FSYNC windows.fsync	Enables stricter scheduling policies to improve frame rates and responsiveness. Can cause issues in certain apps. ⇒ Off 0, On 1.
PBA windows.pba	Implements a GL-free GPU heap allocator, vastly improving the speed of buffer maps. ⇒ Off 0, On 1.
FSR windows.fsr	AMD FidelityFX Super Resolution uses advanced scaling technologies to boost FPS by rendering the game at a lower resolution and then upscaling it. ⇒ Off 0, On 1.
NVAPI windows.nvapi	NVAPI is NVIDIA's core software that allows direct access to NVIDIA GPU. ⇒ Off 0, On 1.
FPS LIMITER windows.fps_limit	Sets the FPS limiter to 60. Required by some older titles. ⇒ Off 0, On 1.
MF windows.mf	Installs Media Foundation, require by some apps which use the Microsoft-specific codecs for audio and video to run. ⇒ Off 0, On 1.
VIRTUAL DESKTOP windows.virtual_desktop	Define the resolution and a new dedicated window. ⇒ Off 0, On 1.
ENABLE MOUSE windows.force_mouse	Show the cursor. ⇒ Off 0, On 1.
ENABLE XIM windows.allow_xim	Enable XIM support. ⇒ Off 0, On 1.
NO WRITE WATCH windows.no_write_watch	Disable support for memory write watches in ntdll. (Dangerous speed hack. Only for specific game engines which don't need them such as CoreRT.) ⇒ Off 0, On 1.
FORCE LARGE ADDRESS windows.force_large_adress	Pretend that 32-bit .exe files have their Large Address Aware flag set. (Grants non-LAA 32-bit games access to as much as 4GiB of RAM instead of 2GiB. May fix OOM crashes or improve performance but could also cause crashes if the game has certain kinds of signed/unsigned sloppiness in its memory handling. Note that upstream Proton considers it safe to have this enabled by default.) ⇒ Off 0, On 1.

ES setting name batocera.conf_key	Description ⇒ ES option key_value
HEAP DELAY FREE <code>windows.heap_delay_free</code>	Delay freeing some memory. (Workaround for games with use-after-free bugs (eg. SimCity for Windows 3.x) which would be applied automatically by Windows's bundled AppCompat database.) ⇒ Off 0, On 1.
HIDE NVIDIA GPU <code>windows.hide_nvidia_gpu</code>	Force Nvidia GPUs to always be reported as AMD GPUs. (For games which would otherwise expect nVidia features too new for the Wine/driver/etc. version Batocera uses.) ⇒ Off 0, On 1.
ENABLE DEBUG <code>windows.wine_debug</code>	Enable wine's debug logging. (Useful for identifying what compatibility settings or winetricks need to be applied or for filing Wine/Proton bug reports.) ⇒ Off 0, On 1.

Advanced options will only be applied on the first launch of the game. Adjusting these values later will not do anything.

If discovered that the wrong settings were used, delete the game's installation from the `saves/windows/<game>` folder, set up the correct settings, and try launching it again.

Folder compression

Installed games can be converted into two other formats: TGZ and SquashFS. Batocera requires changing the extensions to `*.wtgz` and `*.wsquashfs` respectively. This conversion is doable via the command line. For TGZ:

```
batocera-wine windows wine2winetgz xmoto.wine
```

or for SquashFS:

```
batocera-wine windows wine2squashfs xmoto.wine
```

TGZ and SquashFS files cannot be written to (they are read-only). To work around this:

- When a TGZ is detected, it is fully unzipped to the saves directory and a copy is created
- If a SquashFS is detected instead, Batocera will create a new directory in `saves/windows/<game name>/` which saves the difference in files between the read-only SquashFS and what the game writes to its directory during execution.

Ordinarily, Batocera would just let the game directly write data to the `roms/windows/<game name>.wine/` folder when it wants to.

To reiterate, it is preferable to use:

- `wtgz` for small games that make large writes (at least proportionally) where it makes more sense to just quickly repackage the game in order to keep things portable
- `wsquashfs` for larger games that make small writes (such as save files only) where it doesn't make sense to rapidly duplicate it, as that would take too long/wear down the storage hardware

too quickly

This makes it easier to remove and restore a game, as it won't be modified with every launch. Remember to keep track of the save directory as well!

Folder compression using Windows PowerShell Script

User sirMagb has developed a Windows Power-Shell Script for folder compression in Windows.



[wsquashfs_power-shell_script.zip](#)

- Extract the files to a folder.
- Right click on **CompressFolder.ps1** and choose Run with PowerShell.

Creating autorun.cmd from SSH

To create an autorun.cmd you can manually edit each file or you can type `batocera-wine windows autorun <GAME.DIR> <DIR_IN_GAME.DIR>`. If you leave both brackets empty then your current directory is used for **<GAME.DIR>** and as well as **<DIR_IN_GAME.DIR>** - so be warned!

Some examples:

```
batocera-wine windows autorun "Age of Empires.wine"  
batocera-wine windows autorun "Age of Empires.wine" drive_c/P*  
batocera-wine windows autorun
```

To find the correct **exe**-file batocera-wine can be equipped with a text file that contains regular expression (RegEx) and that will help to **exclude some exe**-files. Ideally we want to find only a single exe because this will lead to an automatic generation (it is same process if you install a Windows Application), if not you can select a file out from a list. If everything is gone well and you found some exe-files the output will likely look like this:

```
'Typhoon 2001.pc'  
Windynomate.pc  
'Worms Armageddon.pc'  
Z.pc  
'Zuma Deluxe.pc'  
'Zuma HD Remake (freeware).pc'  
'Zuma Revenge.pc'  
[root@BATOCERA /userdata/roms/windows]# batocera-wine windows autorun Z.pc  
Found 4 files in Z.pc  
1) . Installer.exe  
2) . Loader.exe  
3) . winz.exe  
4) . ZEDITOR.EXE  
Select entry to write to autorun.cmd: █
```

The RegEx list can be placed on two locations and has to be named `autorun-regex.txt`. If you place your strings in both places, only the primary-file one is used! If you want to comment out an expression then use the hashtag literal.

1. `/userdata/system/roms/windows_installers/` (primary)
2. `/userdata/saves/windows_installers/` (secondary)

[| autorun-regex.txt](#)

```

1. # This is user file to create autorun.cmd by WINDOWS_INSTALLERS or
   'batocera-wine windows autorun <GAME_DIR> <PATH_IN_GAME_DIR>'
2. # Add addition strings, avoid trailing spaces you can comment out
   values by using # 23.02.2025 crcerror
3.
4. # Variation of Uninstaller/Install files
5. ^.*/*[sS]etup\.exe$
6. ^.*/*[Ee]ditor.*$
7. ^.*/*[-_\ ][Uu]ninstall\.exe$
8. ^.*/*[Rr]emove\.exe$
9. ^.*/*[Uu]nwise[[:digit:]]{0,2}\.exe$
10. ^.*/*[Uu]ninstall[-_\ ].*\.exe$
11. ^.*/*[Ii]nstall(..)?\.exe$
12.
13. # Supplementary Directories
14. ^.*/*Adobe AIR/*.*$
15. ^.*/*[Uu]install[ers]{0,3}/*.*$
16. ^.*/*InstallShield Installation Information/*.*$
17. ^.*/*InstallShield/*.*$
18. ^.*/*_Redist/*.*$
19.
20. # Supplementary Files
21. ^.*/*dpinst\.exe$
22. ^.*/*[Vv][Cc][\._]?redist[\._]x[86][64]\.exe$
23. ^.*/*[Qq]uick[Ss][Ff][Vv]\.[Ee][Xx][Ee]$
24.
25. # Cracks/Patches
26. ^.*/*no-cd.*$
27. ^.*/*[Cc]rack(..)?/*.*$
28. ^.*/*[Pp]atch(..)?/*.*$

```

To automate the process you can use a small script, that automatically searches your Windows-games and calls the script for every single directory. Existing `autorun.cmd` files will be renamed to `autorun.cmd.bak`, `autorun.cmd.bak.~1~`, etc.

[Autoautorun.sh](#)

```

#!/bin/bash
readarray -t array <<(find /userdata/roms/windows -mindepth 1 -
maxdepth 1 -name "*" -type d)

```

```
for i in "${array[@]"; do
    batocera-wine windows autorun "$i" .
done
```

Launching a Windows game from SSH

For debugging purposes, to launch a Windows game from SSH, use the following commands:

```
export DISPLAY=:0.0
batocera-wine windows play /userdata/roms/windows/Windlands.pc
```

Where /userdata/roms/windows/Windlands.pc is the path to the game

Manually add redistributables, packages and registry entries to your game

Create the folder in the appropriate locations:

Filetype	Install location	Backuplocation
.exe	/userdata/system/wine/exe	/userdata/system/wine/installed.exe
.msi	/userdata/system/wine/msi	/userdata/system/wine/installed.msi
.ttf/.ttc	/userdata/system/wine/fonts	/userdata/system/wine/installed.fonts
.reg	/userdata/system/wine/regs	/userdata/system/wine/installed.regs

Remember: Every installation/adding is logged to /userdata/system/wine/imports.log

Manually install an exe files

To install dependencies inside a WINEprefix, is by creating a folder in /userdata/system/wine/exe, and then putting the required executable files inside. For 32-bit architectures, install the 32-bit version, and for 64-bit architectures, install both the 32 and 64-bit versions. For each one, a download SSH command will be given. Read [here](#) the related licenses for those packages.

Common list of redistributable from Batocera server

- **DirectX3D** : <https://batocera.org/users/liberodark/wine/directx.7z>
 - wget -P /userdata/system/wine/exe <https://batocera.org/users/liberodark/wine/directx.7z> && 7zr x /userdata/system/wine/exe/directx.7z -o/userdata/system/wine/exe && rm /userdata/system/wine/exe/directx.7z

Common list of Visual C++ Redistributable from Batocera server

• **2015-2022 Visual C++ Redistributable :**

- 32-bit: https://batocera.org/users/liberodark/wine/vcredist_x86_2015_2022.exe
 - `wget -P /userdata/system/wine/exe`
`https://batocera.org/users/liberodark/wine/vcredist_x86_2015_2022.exe`
- 64-bit: https://batocera.org/users/liberodark/wine/vcredist_x64_2015_2022.exe
 - `wget -P /userdata/system/wine/exe`
`https://batocera.org/users/liberodark/wine/vcredist_x64_2015_2022.exe`

• **2015-2019 Visual C++ Redistributable :**

- 32-bit: https://batocera.org/users/liberodark/wine/vcredist_x86_2015_2019.exe
 - `wget -P /userdata/system/wine/exe`
`https://batocera.org/users/liberodark/wine/vcredist_x86_2015_2019.exe`
- 64-bit: https://batocera.org/users/liberodark/wine/vcredist_x64_2015_2019.exe
 - `wget -P /userdata/system/wine/exe`
`https://batocera.org/users/liberodark/wine/vcredist_x64_2015_2019.exe`

• **2019 Visual C++ Redistributable :** for missing files ???, ???, or files ending with `vc16.dll`.

- 32-bit: https://batocera.org/users/liberodark/wine/vcredist_x86_2019.exe
 - `wget -P /userdata/system/wine/exe`
`https://batocera.org/users/liberodark/wine/vcredist_x86_2019.exe`
- 64-bit: https://batocera.org/users/liberodark/wine/vcredist_x64_2019.exe
 - `wget -P /userdata/system/wine/exe`
`https://batocera.org/users/liberodark/wine/vcredist_x64_2019.exe`

• **2017 Visual C++ Redistributable :** for missing files `mfc140ud.dll`, `msvcp150.dll`, or files ending with `vc15.dll`.

- 32-bit: https://batocera.org/users/liberodark/wine/vcredist_x86_2017.exe
 - `wget -P /userdata/system/wine/exe`
`https://batocera.org/users/liberodark/wine/vcredist_x86_2017.exe`
- 64-bit: https://batocera.org/users/liberodark/wine/vcredist_x64_2017.exe
 - `wget -P /userdata/system/wine/exe`
`https://batocera.org/users/liberodark/wine/vcredist_x64_2017.exe`

• **2015 Visual C++ Redistributable :** for missing files `mfc140u.dll`, `msvcp140.dll`, or files ending with `vc14.dll`.

- 32-bit: https://batocera.org/users/liberodark/wine/vcredist_x86_2015.exe
 - `wget -P /userdata/system/wine/exe`
`https://batocera.org/users/liberodark/wine/vcredist_x86_2015.exe`
- 64-bit: https://batocera.org/users/liberodark/wine/vcredist_x64_2015.exe
 - `wget -P /userdata/system/wine/exe`
`https://batocera.org/users/liberodark/wine/vcredist_x64_2015.exe`

15.exe

- **2013 Visual C++ Redistributable** : for missing files mfc120u.dll, msvcp120.dll, or files ending with vc12.dll.
 - 32-bit: https://batocera.org/users/liberodark/wine/vcredist_x86_2013.exe
 - wget -P /userdata/system/wine/exe
https://batocera.org/users/liberodark/wine/vcredist_x86_2013.exe
 - 64-bit: https://batocera.org/users/liberodark/wine/vcredist_x64_2013.exe
 - wget -P /userdata/system/wine/exe
https://batocera.org/users/liberodark/wine/vcredist_x64_2013.exe
- **2012 Visual C++ Redistributable** : for missing files mfc110u.dll, msvcp110.dll, or files ending with vc11.dll.
 - 32-bit: https://batocera.org/users/liberodark/wine/vcredist_x86_2012.exe
 - wget -P /userdata/system/wine/exe
https://batocera.org/users/liberodark/wine/vcredist_x86_2012.exe
 - 64-bit: https://batocera.org/users/liberodark/wine/vcredist_x64_2012.exe
 - wget -P /userdata/system/wine/exe
https://batocera.org/users/liberodark/wine/vcredist_x64_2012.exe
- **2010 Visual C++ Redistributable** : for missing files mfc100u.dll, msvcp100.dll, or files ending with vc10.dll.
 - 32-bit: https://batocera.org/users/liberodark/wine/vcredist_x86_2010.exe
 - wget -P /userdata/system/wine/exe
https://batocera.org/users/liberodark/wine/vcredist_x86_2010.exe
 - 64-bit: https://batocera.org/users/liberodark/wine/vcredist_x64_2010.exe
 - wget -P /userdata/system/wine/exe
https://batocera.org/users/liberodark/wine/vcredist_x64_2010.exe
- **2008 Visual C++ Redistributable** : for missing files mfc90u.dll, msvcp90.dll, or files ending with vc9.dll.
 - 32-bit: https://batocera.org/users/liberodark/wine/vcredist_x86_2008.exe
 - wget -P /userdata/system/wine/exe
https://batocera.org/users/liberodark/wine/vcredist_x86_2008.exe
 - 64-bit: https://batocera.org/users/liberodark/wine/vcredist_x64_2008.exe
 - wget -P /userdata/system/wine/exe
https://batocera.org/users/liberodark/wine/vcredist_x64_2008.exe
- **2005 Visual C++ Redistributable** : for missing files mfc80u.dll, msvcp80.dll, or files ending with vc8.dll.
 - 32-bit: https://batocera.org/users/liberodark/wine/vcredist_x86_2005.exe
 - wget -P /userdata/system/wine/exe
https://batocera.org/users/liberodark/wine/vcredist_x86_2005.exe
 - 64-bit: https://batocera.org/users/liberodark/wine/vcredist_x64_2005.exe

- `wget -P /userdata/system/wine/exe https://batocera.org/users/liberodark/wine/vcredist_x64_2005.exe`

Once placing the required *.exe-files for the game as described previously, launch the game, and the dependencies will install themselves silently. After closing the game, the deployed redistributable files will be moved to `/userdata/system/wine/installed.exe/` folder. Batocera will never try to reinstall the files again, when the game is started again. You can follow previous installs if you open `/userdata/system/wine/imports.log`, there you find a log of all installed packages. Place only *.exe-files in the folder named above.

Manually install a msi file

To use a Windows Installer package inside your WINEprefix, create a folder in `/userdata/system/wine/msi`, and then putting the required msi-package inside. Start the game and the dependencies will install themselves. Place only *.msi-files in the folder named above. After finishing the game you will find the used msi-packages in `/userdata/system/wine/installed.msi` folder.

Manually put a registry file

To apply specific registry values for a given game, create a folder in `/userdata/system/wine/regs/` and put the registry files (*.reg) inside. Once the game is launched, it will import the values contained in this file. Place only *.reg-files in the folder named above. Once the game is closed, the used file can be found in `/userdata/system/wine/installed.regs/`. Batocera will therefore never try to re-apply those files when the game is started.

Not recommended is to use `/var/run/rawinput.reg`, this is for developers and after import the file is deleted.

Manually put a font file

You can put font-files to folder `/userdata/system/wine/fonts`, the extensions *.ttf and *.ttc are accepted. After the game is started these files will be copied to to Wine-windows-fonts-directory. The installed extensions can be found in `/userdata/system/wine/installed.fonts`

Advanced

This section talks about some advanced topics:

1. WINE DLL Override Instructions
2. Using WINetricks

DLL Override Instructions

This write-up is an excerpt about handling DLL Overrides from Batoceras WINE. Let us start with some background information about DLLs.

A Dynamic Link Library (DLL) file contains compiled code, data, and resources that can be shared by multiple applications. On Windows, DLL files are typically stored in system folders, but game developers sometimes modify them by adding custom code. These modified DLLs are then placed in the game's folder with the same name as the original system DLL. By default, Windows (the OS) prioritizes the DLL located in the game's folder over the one in the system directory. However, **WINE reverses this behavior:**

- it looks for **DLL files in its system directories** at **first**
- then in the **game's folder**, in **second**.

This means WINE may use the default version of a DLL even if a customized version exists in the game's directory.

To change this behavior in Wine, you can use a **DLL Override**. There are several ways to do this, including through the terminal, but one of the more user-friendly methods is editing the `autorun.cmd` by adding the using the ENV parameter. You can also edit the `user.reg` file using Notepad++ over a network connection.

Adding the ENV parameter

1. First, add your game to the Batocera system, launch it once and then exit the game back to ES, which will create the required WINE bottle.
2. Edit the `/userdata/roms/windows/<yourGame>.pc/autorun.cmd` file and add the according `ENV=WINEDLLOVERRIDES=<NameOfDllFile>="native"`
 1. Do not use the dll-extension, just the plain filename
 2. `native` means to override the DLL
 3. `builtin` use the system DLL
 4. `disabled` does not load the DLL
3. Save the file and start the game

*You can daisychain WINEDLLOVERRIDES like

```
ENV=WINEDLLOVERRIDES="msvbvm50,comctl32.ocx=n"
```

```
or ENV=WINEDLLOVERRIDES="NameOfDllFile1"="native"
```

```
WINEDLLOVERRIDES="NameOfDllFile2"="native"*
```

Editing the user.reg file

1. First, add your game to the Batocera system and launch it once. This will create the required Windows file structure.
2. On a PC connected to the same network as your Batocera system, navigate to:
`/share/system/wine-bottles/windows/ge-custom/<game>.pc.wine`
3. Locate the `user.reg` file in that directory. Right-click it and choose Edit with Notepad++.
4. Press `Ctrl + F` and search for: `[Software\\Wine\\DllOverrides]`
5. Add your override in this section:
 - **Here's how you can format your override entries:**
 - `<NameOfDllFile1>="native,builtin"`
 - `<NameOfDllFile2>="native"`

- “<NameOfDllFile3>”=“builtin”
- “<NameOfDllFile4>”=“disabled”
- ...and so on.

Remember, you **DO NOT want to include the file extension in your entry**. For example, if you're overriding ddraw.dll, then you would simply say ddraw before the equal sign. If you're unsure which to use, start with “native, builtin” and test other options if necessary.

WINEtricks

Some games require tricks to run. For example, Age of King requires directplay for network play (<https://appdb.winehq.org/objectManager.php?sClass=version&iid=147>). It can be installed automatically thanks to the tricks command.

```
export DISPLAY=:0.0
batocera-wine windows tricks /userdata/roms/windows/aok.wine directplay
```

A list of WINEtricks commands and more detail about using WINEtricks with Batocera is available [right here](#).

If a game needs several tricks you can add as many trick commands as you like (since v43), so you don't need to call the command every time

```
batocera-wine windows tricks /userdata/roms/windows/aok.wine directplay
vb5run vb6run
```

Troubleshooting

Down here there are some tricks to achieve a good running on games. It's better to try some procedures written here and then later come back with your findings on discord. A message like **My game isn't working** will not help anyone. Best feedback can be given, if you provide logs (found in /userdata/system/logs)

My game isn't working!

It can be very hard to troubleshoot what exactly is going wrong for games that are failing to run in WINE. First, check that it's on the spreadsheet as confirmed working before continuing troubleshooting, it might just be “borked” and cannot be fixed. Also, check the [WineHQ application database](#).

Next, try launching the game via SSH, as that will display some output in regards to the error it's encountering. This could lead to discovering which WINEtrick is required.

After discovering which [advanced system setting](#) adjustment got the game working, be sure to delete its saves/windows/<game> folder before trying again.

My application isn't working!

As above, but here there may be more options. Check if the distributor/creator of the program offers a “portable” version, one that doesn't require writing files to the system directories and keeps everything self-contained. Some programs can be told to behave in a portable-like manner by having a `portable.txt` file present in their directory. Of course, for programs set up like this, do not use SquashFS, as games set up in this way require being able to write into their installed directory instead.

Sophisticated applications such as ones that require low-level access to hardware (such as drivers, patchers, kernels, complicated professional production software like compilers, etc.) may just not work due to the nature of WINE.

My game/app was working before but now is no longer!

Try deleting the `/userdata/system/Wine/` folder, then try running the game again. Wine may need to reinstall, so be patient.

Which WINE versions are installed on Batocera?

Knowing which versions of WINE is actually installed on Batocera is essential for troubleshooting.

From v39, Batocera switch to one runner, the Glorious Eggroll runner. Execute the following commands from a remote PC via SSH to get the version:

- Get WINE version for Wine-GE:

```
/usr/wine/ge-custom/bin/wine64 --version
```

For previous Batocera versions, to display the WINE versions for Lutris and Proton, execute the following commands from a remote PC via SSH:

- Get WINE version for Lutris:

```
/usr/wine/lutris/bin/wine64 --version
```

- Get WINE version for Proton:

```
/usr/wine/proton/bin/wine64 --version
```

How to install custom WINE versions/runners in Batocera



With v40 and later, you can now install a compatible version of Wine into the folder `/userdata/system/wine/custom/`. The runner should be in it's own folder, i.e. `/userdata/system/wine/custom/Wine-9.4/`



A good resource for WINE-versions: <https://github.com/Kron4ek/Wine-Builds>

1. Create the directory

```
/userdata/system/wine/custom
```

if it doesn't already exist. This can be done using Windows/Samba or Terminal/WinSCP.

2. Download the desired WINE-version and unzip and copy them into its own directory like

```
/userdata/system/wine/custom/Wine-9.4
```

This can be done using Windows/Samba or WinSCP.

3. Using the Terminal or SSH, navigate to `/userdata/system/wine/custom` using

```
cd /userdata/system/wine/custom
```

4. Set the correct rights for execution of the WINE files using

```
chmod a+x -R *
```

5. Batocera EmulationStation will then allow you to choose the runner of your choice and you should be ready to go.



Further troubleshooting

For further troubleshooting, refer to the [generic support pages](#).

From:

<https://www.wiki.batocera.org/> - **Batocera.linux - Wiki**

Permanent link:

<https://www.wiki.batocera.org/systems:windows>

Last update: **2026/01/22 10:28**

