# Storage Manager

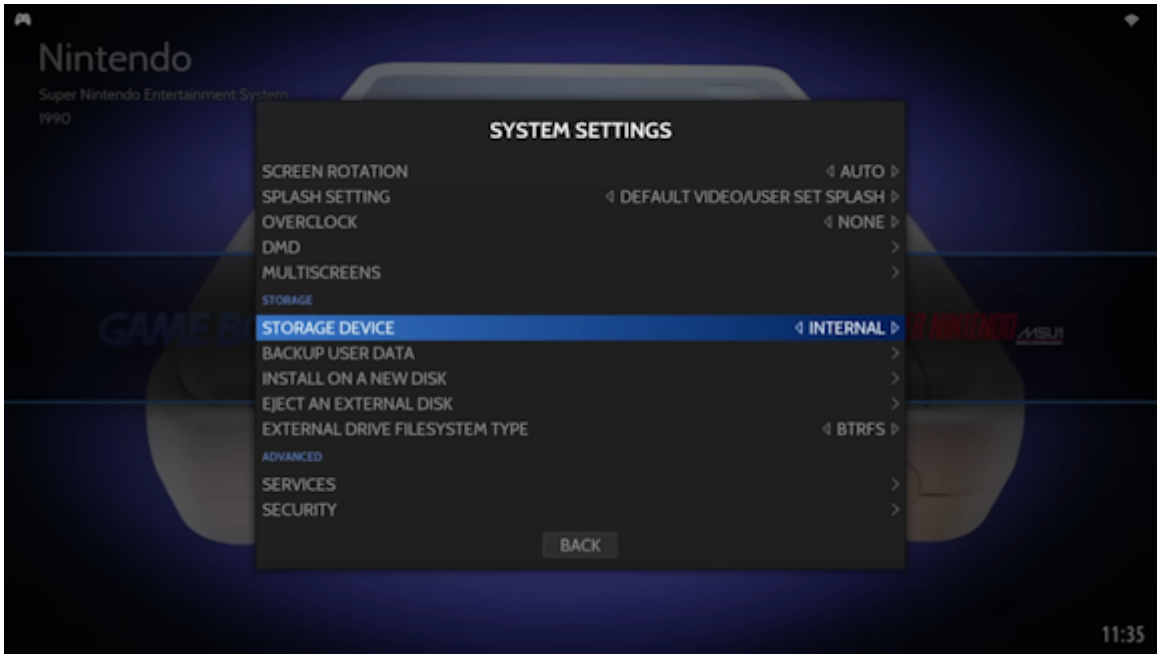
> Introduced in **Batocera v43**.

Batocera's **Storage Manager** allows you to expand your ROM storage across multiple drives while keeping a single unified game library.

USB drives, external hard drives, SSDs, SD cards, and additional internal drives can all be used.

The Storage Manager introduced in Batocera v43 allows multiple locally attached drives to be combined into a single ROM library without requiring symbolic links, manual mounts, or custom scripts.

Advanced users can also create manual MergerFS pools that include network/NAS mounts. This is configured in `/boot/batocera-boot.conf` and is not managed by the Storage Manager user interface. See Network/NAS Storage (Advanced).




## Overview

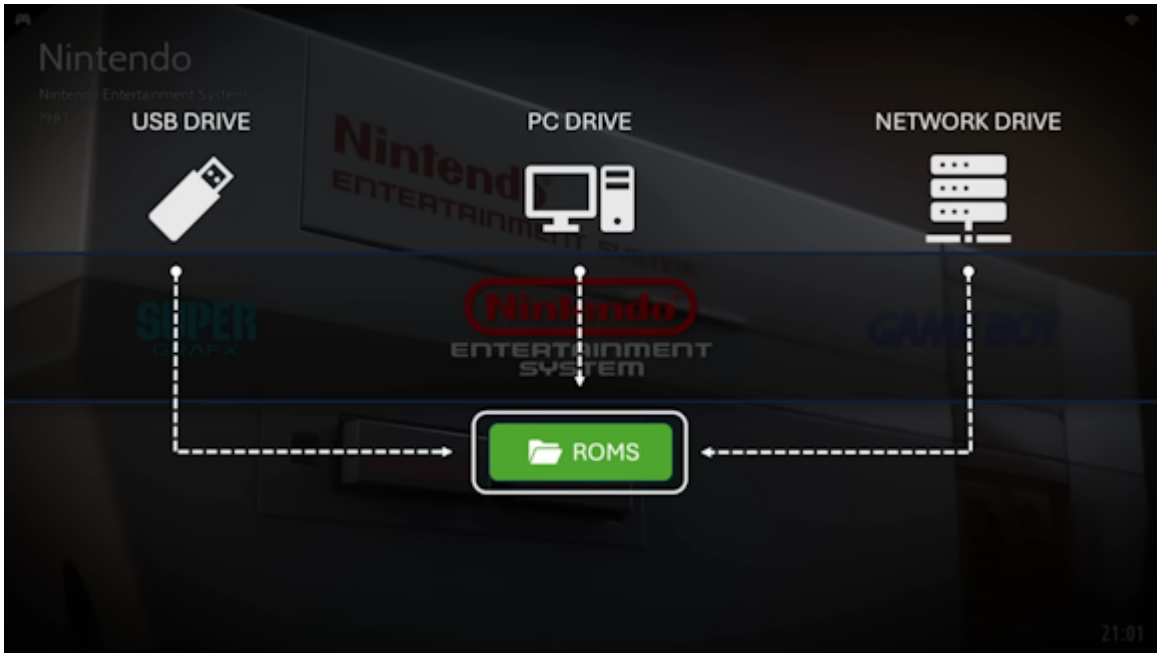
The Storage Manager can:

- Detect compatible storage devices automatically
- Merge existing ROM drives
- Format and prepare new drives
- Use multiple drives simultaneously
- Present all ROMs as a single library under `/userdata/roms`
- Safely eject drives
- Automatically refresh game lists

From the user's point of view, there is still only one ROM folder:

```
/userdata/roms
```

Batocera handles the storage pool in the background.




A simplified overview:

```
Internal ROMs
     |
Drive 1 /roms
     |
Drive 2 /roms
     |
  MergerFS
     |
/userdata/roms
```

## Which storage method should I use?

Batocera has several storage mechanisms that solve different problems. They should not be treated as interchangeable.

| Method | Best used for | What it does |
|---|---|---|
| **Storage Manager** | Multiple locally attached ROM drives | Builds a MergerFS union at `/userdata/roms` and manages it from EmulationStation |
| **sharedevice=DEV / ANYEXTERNAL** | Moving all userdata to one selected device, or to the first suitable external device | Mounts another device as the active userdata storage |

| Method | Best used for | What it does |
|---|---|---|
| **sharedevice=DEVICES** | Mounting selected local partitions or subdirectories over specific userdata folders | Bind-mounts selected local storage over individual targets such as ROMS, SAVES or BIOS |
| **sharedevice=NETWORK** | Replacing all or selected userdata folders with SMB/NFS shares | Mounts network storage directly on the requested userdata paths |
| **Manual S12mergerfs pool** | Combining local folders and network/NAS folders in one union | Builds an advanced MergerFS pool from explicitly configured branches |

If the goal is simply to add a second, third or fourth locally attached ROM drive, use the graphical Storage Manager. If the goal is to combine a NAS with local ROM storage in the same `/userdata/roms` namespace, use the advanced manual MergerFS method documented below.

# Requirements

For drives detected and managed automatically by the Storage Manager user interface, the drive must contain a root-level folder named:

```
roms
```

Supported:

```
Drive
└── roms
    ├── nes
    ├── snes
    └── psx
```

Not supported:

```
Drive
└── Batocera
    └── roms
```

Not supported:

```
Drive
├── nes
├── snes
└── psx
```

The Storage Manager user interface currently expects the `roms` folder to exist directly in the root of the mounted drive.

This root-level folder requirement applies to automatic Storage Manager detection. Advanced manual MergerFS pools can use other mounted directory paths, including nested local folders and network

shares.

### Existing disk layout and filesystems

Automatic Storage Manager detection is not limited to disks that have a conventional partition table. The current backend handles both of these cases:

- A partitioned disk, where each partition is detected and mounted separately
- A whole disk containing a filesystem directly, with no partition table, which can be mounted as a raw filesystem

In either case, automatic merging still requires a `roms` directory at the root of the mounted filesystem.

An existing partition that has no recognizable filesystem is **not** treated the same way as a completely blank disk. The automatic **Format & Merge** prompt is generated by the whole-disk empty-disk path after Batocera has checked for a partition table, partitions and a raw filesystem. Therefore, a disk containing an unusable or unformatted partition may need to be repartitioned or formatted separately before Storage Manager can use it.

## Using Existing Drives

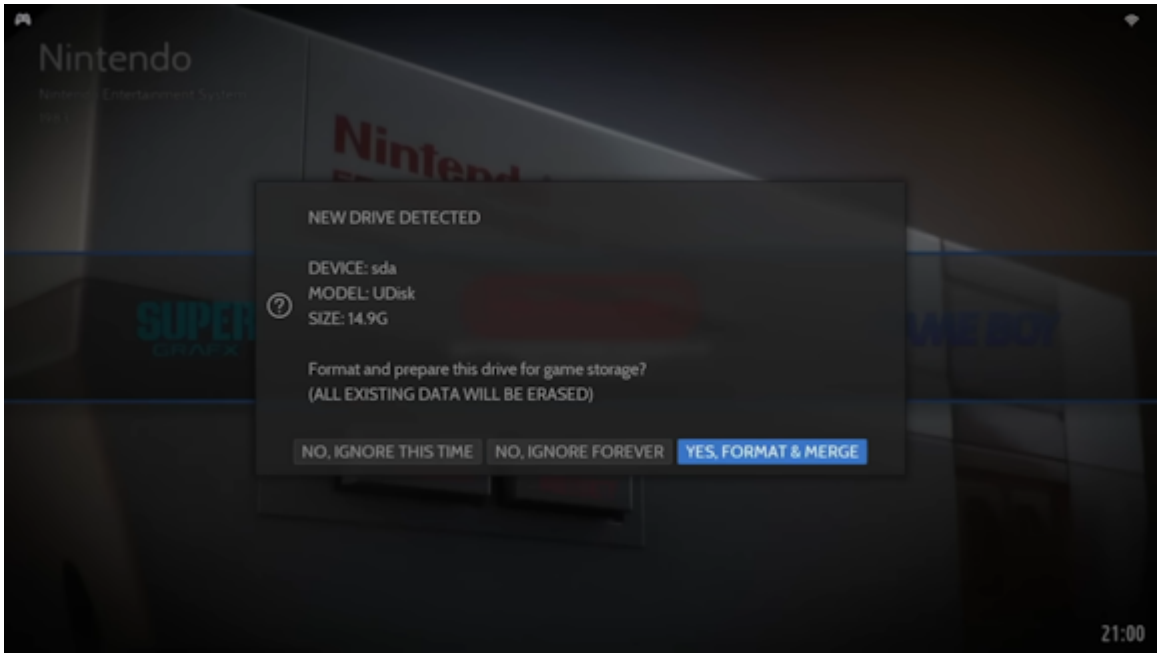
When Batocera detects a mounted drive containing a compatible root-level `roms` folder, it offers to merge the drive into the Storage Manager.

After accepting the prompt, Batocera adds the drive to the storage pool.

After the merge completes, the ROMs stored on the drive become available alongside your existing collection.

## Formatting a New Drive

If a blank drive is detected, Batocera can automatically prepare it for game storage.




Formatting a drive will permanently erase all existing data stored on that drive. Make sure any important files have been backed up before proceeding.

Selecting **Format & Merge** will format the drive and add it to the Storage Manager.

Batocera will:

- Format the drive
- Create a root-level `roms` folder
- Generate the standard ROM directory structure
- Merge the drive into the Storage Manager
- Refresh the game lists

## Filesystem choices and what formatting does

The Storage Manager delegates disk formatting to `batocera-format`. In the implementation documented here, the available format types are:

```
ext4
btrfs
```

```
exfat
```

If no filesystem type is supplied to the backend, `ext4` is the default.

Formatting operates on the **whole selected disk**, not only on an existing partition. Batocera erases the beginning of the disk, creates a new GPT partition table, creates one partition using the disk, formats it, and labels the new filesystem ROMS. All existing partitions and data on that disk can therefore be lost.

After formatting, Batocera creates the root-level `roms` directory. It then copies the **directory structure only** from the existing ROM library to the new drive before merging it. Existing ROM files are not duplicated to the newly formatted drive by this step.

Existing drives do not need to use one of the three formatting choices above if Batocera can already mount their filesystem. NTFS, for example, has explicit mount handling in the Storage Manager, but NTFS is not offered by `batocera-format` as a format target in this implementation.

For an existing NTFS filesystem, Storage Manager currently runs `ntfsfix -d` first, then tries the kernel `ntfs3` driver. If that mount fails, it falls back to `ntfs-3g`. Other recognized filesystems are normally mounted using the filesystem detected by `blkid` with `noatime`.

The backend also independently refuses a format request if the selected device resolves to Batocera's system disk. When formatting proceeds, `batocera-format` creates one GPT partition beginning at 2 MiB and extending to the end of the disk.

## Using Multiple Drives

Multiple drives can be used simultaneously.

Example:

```
/media/roms1/roms
/media/roms2/roms
```

Batocera combines them into a single virtual library:

```
/userdata/roms
```

Games can be distributed across multiple drives while still appearing as one collection inside EmulationStation.

## Network/NAS Storage (Advanced)

The graphical Storage Manager currently focuses on locally attached storage devices. Batocera also includes a separate boot-time MergerFS service that can build manual pools from local storage, removable media and already-mounted network shares.

This makes it possible to present ROMs from a NAS together with ROMs stored on local drives under the same virtual directory, such as:

```
/userdata/roms
```

Manual MergerFS pools are an advanced configuration and are not managed by the Storage Manager user interface. If you create a manual pool for `/userdata/roms`, do not also use the graphical Storage Manager to manage that same pool. The graphical Storage Manager owns its own `mergerfs.roms` configuration and can rebuild the live ROM pool independently.

## Manual MergerFS Syntax

Manual pools are configured in:

```
/boot/batocera-boot.conf
```

The basic syntax is:

```
mergerfs=<pool>@<branch1>:<branch2>:<branch3>
```

For example:

```
mergerfs=/userdata/roms@/media/BATO_HDD/roms:/media/SDCARD/path/to/roms
```

The path before the @ is the virtual MergerFS mount point. Paths after the @ are the storage branches that are combined into the pool.

Unlike automatic Storage Manager detection, manual branches do not need to use a root-level `roms` directory. Any suitable mounted directory can be used as a branch.

The manual service accepts more than one pool. Keys beginning with `mergerfs` followed by letters, numbers, underscores or hyphens are processed, for example:

```
mergerfs=/userdata/roms@/media/SSD/roms:/media/NAS/roms
mergerfs_bios=/userdata/bios@/media/SSD/bios:/media/NAS/bios=RO
```

The manual parser uses @ to separate the pool from its branch list and `:` to separate branches. Avoid using those characters inside local branch paths. Mount network shares to a simple local path such as `/media/NAS` first, then use that local path as the branch.

The graphical Storage Manager key is `mergerfs.roms`. The dot in that key means it is not consumed by the manual `S12mergerfs` key parser. They are separate implementations even though both use MergerFS.

## Mounting a NAS Before the Pool Is Created

The manual MergerFS service supports `premerge` commands. These commands run before the MergerFS pool is created and can be used to mount a network share first.

Example using an SMB/CIFS guest share:

```
sharewait=30
premerge1=mkdir -p /media/NAS && mount.cifs //192.168.1.10/Games /media/NAS
-o guest
mergerfs=/userdata/roms@/media/BATO_HDD/roms:/media/SDCARD/path/to/roms:/med
ia/NAS/roms=RO
```

In this example:

- `premerge1` creates the local mount point and mounts the SMB share
- `sharewait` is used by the boot service as the retry budget for `premerge` commands
- `/media/NAS/roms` is added as another MergerFS branch
- =RO makes only that branch read-only inside the MergerFS pool

The example above uses a guest SMB share for simplicity. Password-protected SMB shares and NFS mounts can also be mounted with normal Linux mount options before being included as a MergerFS branch. See the Batocera NAS documentation for the existing SMB/NFS syntax and network-storage guidance.

Multiple `premerge` commands are supported. The service processes all keys matching names such as `premerge1=`, `premerge_nas=`, or similar before it processes any manual MergerFS pools.

If `sharewait` is not set, the service uses a default value of 15. `premerge` retries are attempted at 4-second intervals and the number of attempts is derived from `sharewait / 4` using integer arithmetic. The default value therefore gives three attempts, while `sharewait=30` gives seven attempts. Values below 4 result in **zero** `premerge` attempts, so do not set `sharewait` below 4 when using `premerge`. The same `sharewait` setting is also used by Batocera's normal share service, so changing it can affect both storage mechanisms.

If a `premerge` command exhausts its retries, the current `S12mergerfs` script logs that it is giving up but then continues to the manual pool-processing stage. A NAS mount failure therefore does not itself guarantee that pool creation will be aborted. When troubleshooting a network-backed pool, verify that the expected path is actually mounted before assuming the MergerFS pool is healthy.

This matters especially when the local mount-point directory already exists. If the network mount failed but a branch path still exists locally, MergerFS can see the local directory rather than the NAS. Marking a network branch =RO is a useful additional safeguard when the NAS is intended to be read-only, but it does not replace checking the mount with `findmnt`.

`premerge` values are executed by the boot service through the shell using `eval`. Treat every `premerge` line as a privileged shell command, not as a restricted mount-only setting. Quote paths/options carefully and do not copy untrusted command text into `/boot/batocera-boot.conf`. Credentials written directly into a `mount.cifs` command will also be stored as plain text in that configuration file.

`premerge` runs at boot. It is not a continuous NAS availability monitor. If a network share disappears later, the graphical Storage Manager does not reconnect or eject that network branch for you.

## Branch Access Modes

The manual MergerFS service accepts branch suffixes such as:

```
=RO
=RW
=NC
```

- `RO` = read-only. The branch is excluded from create and action policies.
- `RW` = read/write. This is normal branch behavior.
- `NC` = no-create. The branch is excluded from create policies, but existing files on that branch can still be changed or deleted. `NC` is **not** the same as read-only.

For example:

```
mergerfs=/userdata/roms@/media/SSD/roms:/media/NAS/roms=RO
```

allows ROMs on both branches to appear under `/userdata/roms`, while preventing writes to the NAS branch through the MergerFS pool.

## Manual Pool File Placement

The boot-time manual MergerFS service currently uses:

```
category.create=mfs
moveonenospc=true
minfreespace=4G
```

`mfs` means **Most Free Space**. New files are created on an eligible writable branch with the most available free space.

The complete option set currently passed by `S12mergerfs` is:

```
cache.files=off
dropcacheonclose=false
category.create=mfs
allow_other
use_ino
moveonenospc=true
minfreespace=4G
xattr=passthrough
```

When a manual pool is created, Batocera also creates a hidden local base directory derived from the pool path. For a pool at:

```
/userdata/roms
```

the hidden base is:

```
/userdata/.roms
```

Existing files from the original pool directory are moved into this hidden base before the MergerFS mount is created. The base is then added to the pool with =NC, so new files are not created there by MergerFS. Because NC is no-create rather than read-only, existing files in this hidden base can still be modified or deleted through the pool.

The script also **attempts** to remount the hidden base read-only before starting MergerFS, but that command explicitly ignores failure. Do not rely on the underlying hidden base being read-only. The dependable MergerFS-side rule in the command line is =NC, which prevents creation but is not equivalent to =RO.

The service derives this hidden base path from the final component of the pool path. There is an implementation quirk worth knowing about: the manual pool **key** parser permits digits 0 through 9, but the current hidden-base derivation expression only accepts letters, `0`, `1`, underscores and hyphens in the final pool-directory name. Standard names such as `/userdata/roms`, `/userdata/bios` and `/userdata/saves` are safe. Until that code is corrected, avoid custom manual pool names whose final directory component contains digits 2 through 9 or other unusual characters.

The manual `S12mergerfs` service and the graphical `batocera-storage-manager` use different configuration paths and different MergerFS policies. The graphical Storage Manager uses `mergerfs.roms` and `category.create=eplfs`. The manual boot-time service processes keys such as `mergerfs=`, `mergerfs1=`, or other supported `mergerfs`-prefixed names and currently uses `category.create=mfs`.

There is also a source-level limitation in the current `S12mergerfs` stop path: the stop handler checks $POOL before the configured-pool loop has assigned it. It therefore does not appear to enumerate and unmount every configured manual pool when invoked as a standalone stop action. Treat this as an implementation issue, not as a supported way to manually dismantle all pools. Normal system shutdown/reboot has broader filesystem teardown, or an advanced user can explicitly unmount the pool target they created.

## Relationship to sharedevice=NETWORK and sharedevice=DEVICES

The existing `S11share` mechanism and the manual `S12mergerfs` mechanism solve different problems.

- `sharedevice=NETWORK` mounts SMB/NFS storage **over** selected userdata paths such as `/userdata/roms`. It does not union that directory with the local ROM directory.
- `sharedevice=DEVICES` similarly mounts selected local partitions or subdirectories over named userdata targets.
- The normal share service recognizes targets including `SHARE`, `ROMS`, `SAVES`, `BIOS`, `MUSIC`, `DECORATIONS`, `SCREENSHOTS`, `THEMES`, `CHEATS`, `SOUNDS`, `LIBRARY`, `SPLASH`, `CONFIGS` and

RECORDINGS.
- S11share also accepts advanced sharenetwork_cmdN= entries containing arbitrary shell commands. These belong to the older direct-share mechanism and do not create a MergerFS union by themselves.
- S12mergerfs combines multiple already-available paths into one union mount.

S11share runs before S12mergerfs in the boot sequence. The normal network-share service can therefore be useful for other userdata folders, while a manual MergerFS pool is used for ROMs, but do not configure two different mechanisms to mount directly onto the same target path.

The Storage Manager also deliberately skips local disks that are already referenced by UUID through manual sharedevice… entries, which helps prevent it from automatically taking ownership of manually configured local storage.

The normal S11share service also runs /boot/preshare.sh start when that file exists. Batocera's source specifically notes this hook can be used for tasks such as waking a NAS. Because S11share runs before S12mergerfs, this is another way to prepare infrastructure needed by a later manual MergerFS pool, although premerge remains the pool-specific mechanism.

### Boot ordering and delayed Storage Manager detection

Storage devices can appear very early during boot, before /userdata and the normal share configuration are ready. Batocera coordinates this with the S11share stage instead of blindly rebuilding the ROM pool immediately.

At the end of S11share, Batocera:

- Replays Storage Manager device events that were delayed until the share layer became ready
- Checks for /tmp/mergerfs.pending and runs batocera-storage-manager refresh_pool when required
- Creates /var/run/batocera.share.mounted to signal that normal hotplug processing can proceed

This explains why an already-configured ROM drive detected during early boot can be queued first and only joined to the live pool after the base /userdata storage has finished mounting.

## Internal Drives

The Storage Manager is not limited to removable USB devices.

Additional internal drives can also be used.

For example:

```
/media/roms1/roms
/media/roms2/roms
```

As long as the drive is mounted by Batocera and contains a root-level roms folder, it can be merged into the Storage Manager.

This includes:

- SATA hard drives
- SATA SSDs
- NVMe SSDs
- USB storage devices
- SD cards

The drive name does not matter. What matters is that the mounted drive contains a root-level `roms` folder.

## Automatic detection and safety filtering

The Storage Manager backend watches the common Linux block-device families used by Batocera, including `sdX`, NVMe and MMC devices. Its explicit initial-scan and ignore re-detection loops enumerate those device families, so unusual block-device classes are better handled through the manual storage mechanisms unless they are known to trigger the Storage Manager through another Batocera path. The backend also contains several safety filters intended to avoid treating system, firmware and recovery storage as game disks.

Among other checks, the current implementation:

- Skips the disk backing `/boot` or `/userdata`
- Skips EFI System, Microsoft Reserved and Windows Recovery partition GUIDs
- Skips partitions marked hidden or no-automount
- Filters many Android/Qualcomm firmware and system partition labels
- Does not offer the automatic blank-disk format prompt for disks of 4 GiB or smaller
- Ignores swap partitions
- Skips local disks already claimed by UUID-based `sharedevice…` manual configuration

These checks are safeguards, not a substitute for verifying the disk shown by a format prompt. Always confirm the device and capacity before formatting.

Some of the protection checks intentionally use broad filesystem-label and partition-label matches. External media using names associated with system partitions, for example `userdata`, `recovery`, `vendor` or `persist`, can therefore be skipped by automatic detection. If a perfectly usable external partition does not appear in Storage Manager, its filesystem label and partition label are worth checking in `/var/log/batocera-storage.log`.

## Ignoring a drive

The Storage Manager backend supports an `ignore.drives` list in `/boot/batocera-boot.conf`. Ignoring a drive disables Storage Manager automation for that drive. It does not necessarily prevent Batocera from mounting a readable partition.

If an ignored drive contains a root-level `roms` directory, the drive can remain mounted under `/media`, but Storage Manager will skip adding it to the ROM pool or prompting to merge it. Blank ignored drives also do not trigger the format prompt.

The ignore entry is based on a persistent **whole-disk** identifier when one is available, rather than only on a temporary `/dev/sdX` name. In practice, ignoring a physical disk disables Storage Manager automation for its partitions as well.

The current backend provides an `ignore` action but no matching `unignore` action. The persistent state is simply the space-separated `ignore.drives` value in `/boot/batocera-boot.conf`. To reverse an ignore, that stored identifier has to be removed from the configuration and the drive re-detected, reconnected or detected again after a reboot.

### Mount-point naming

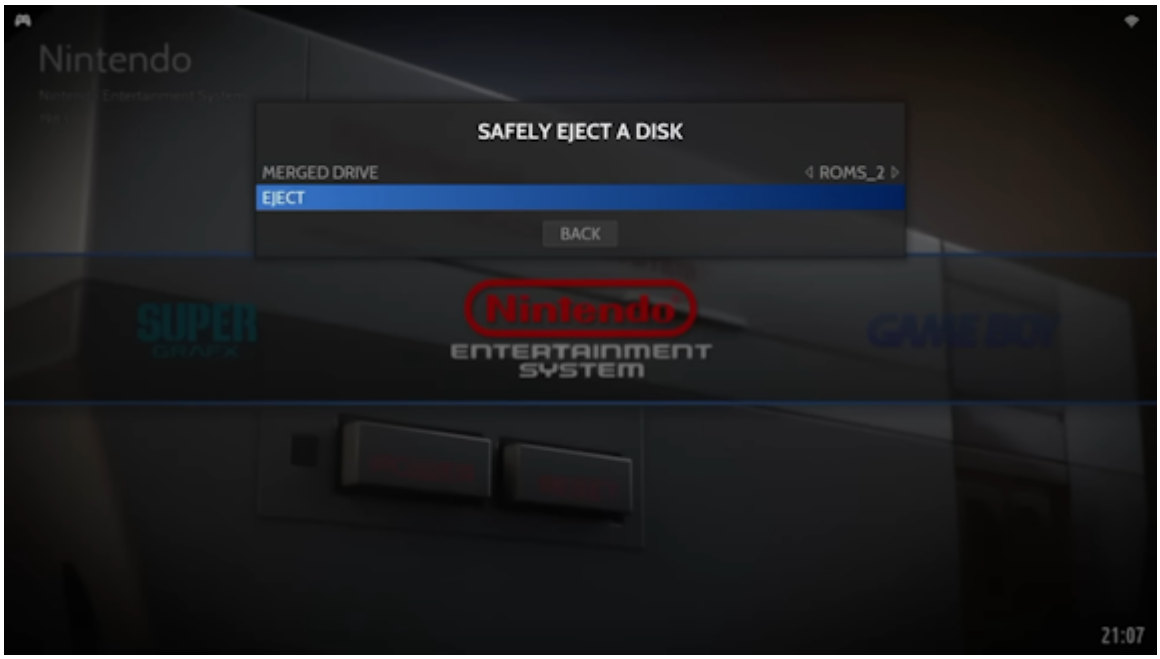
For automatically mounted partitions, Storage Manager builds a mount point under `/media`. If the filesystem has a label, the sanitized label is used; otherwise the partition device name is used. If that mount point already exists, a numeric suffix such as `_1` is added.

Use unique filesystem labels for ROM drives. The Storage Manager configuration stores branch paths such as `/media/MyDrive/roms`. Duplicate labels can produce suffixed mount points and make path identity less predictable across reconnects.

## Safe Drive Removal

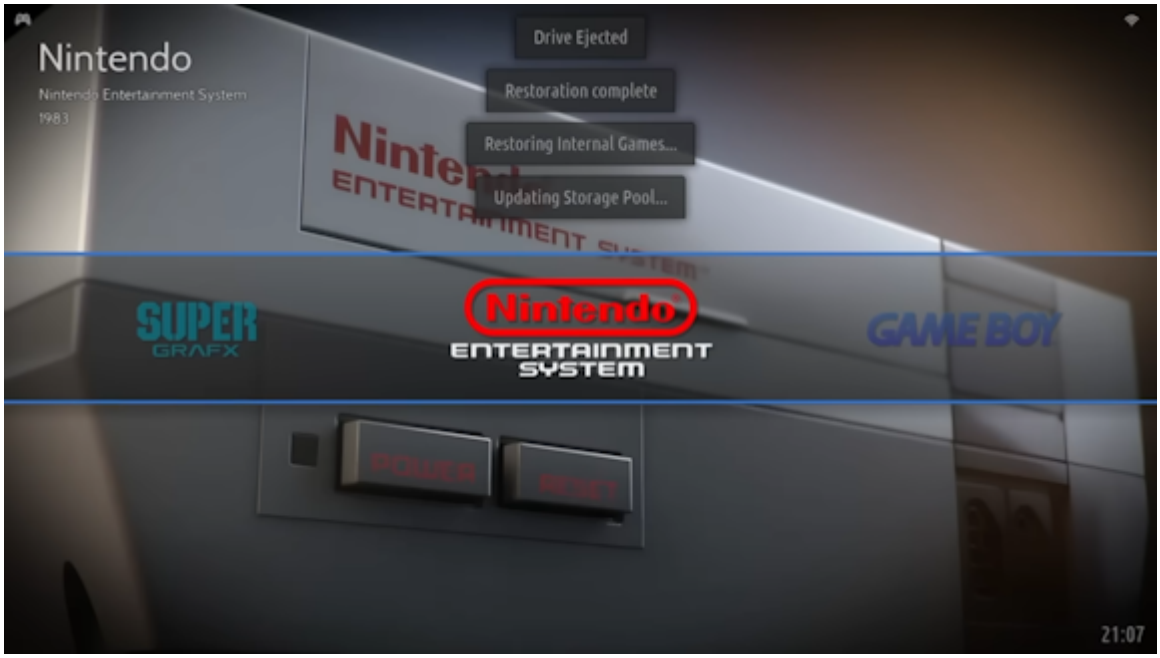
Drives can be safely removed from:

- **Main Menu**
- **System Settings**
- **Eject an Extra Disk**




After selecting the drive, Batocera asks for confirmation before removing it from the storage pool.

Batocera automatically updates the storage pool and refreshes the game lists.




Games stored only on the removed drive will disappear from the library until that drive is connected and merged again.

The **Eject an Extra Disk** list is generated from branches saved in `mergerfs.roms`. It is therefore a list of drives currently managed as Storage Manager pool branches, not a generic eject list for every filesystem mounted under `/media`.

Prefer the Storage Manager eject action over physically unplugging a drive while Batocera is running. The backend has cleanup logic for an unexpected removal, but active writes cannot be made safe after the device has already disappeared.

### Missing at boot, safe eject, and live unplug are different

The Storage Manager keeps configured branches in `mergerfs.roms`. During a pool refresh, a branch whose directory is currently missing is skipped from the live pool rather than automatically deleted from the saved configuration. If that same drive later appears again at the same mount path, Storage Manager can refresh the pool and re-add it automatically.

By contrast, using **Eject an Extra Disk** removes that branch from `mergerfs.roms`. A live unplug event is also routed through the eject/cleanup path when Batocera can identify the removed mount. In those cases, reconnecting the drive requires it to be merged again.

This distinction is useful for systems where a configured drive is sometimes powered off before Batocera boots. Leaving it configured but absent at boot is different from explicitly ejecting it.

## What Gets Merged?

The graphical Storage Manager merges only the ROM library:

```
/userdata/roms
```

Folders such as:

```
/userdata/bios
/userdata/saves
/userdata/screenshots
/userdata/themes
/userdata/music
```

remain on internal storage.

The graphical Storage Manager does not merge BIOS files, save files, screenshots, themes, or music.

Advanced manual MergerFS pools are separate from the graphical Storage Manager and can technically target other directories, but those pools are not managed by the Storage Manager interface.

## Internal ROM Preservation

When the first additional ROM drive is merged, Batocera preserves the existing internal ROM collection.

Internally, existing ROMs stored in:

```
/userdata/roms
```

are moved into:

```
/userdata/.roms_base
```

This allows internal ROMs to remain visible alongside ROMs stored on merged drives.

If all merged drives are later removed, Batocera restores the internal ROM library back to the normal `/userdata/roms` location.

During restoration, the current implementation uses `rsync –ignore-existing`. Existing destination files are therefore not overwritten by files of the same name from `.roms_base`. If restoration fails, Batocera keeps `.roms_base` instead of deleting it so the preserved data is not discarded.

# Limitations

Automatic detection by the graphical Storage Manager currently requires this layout:

```
<drive>/roms
```

Nested ROM folders are not detected automatically.

Example:

```
<drive>/Batocera/roms
```

is not automatically merged.

This limitation does not apply to advanced manual MergerFS branches, where an explicit nested path can be configured.

The graphical Storage Manager merges only the ROM library. Other userdata folders remain independent unless they are configured separately using advanced manual storage methods.

# Advanced Technical Details

## Graphical Storage Manager Pool

Batocera's graphical Storage Manager uses **MergerFS** to combine ROM folders from multiple drives into a single virtual directory.

The merged library is presented at:

```
/userdata/roms
```

Internally, Batocera maintains a storage pool containing:

```
/userdata/.roms_base
```

plus any merged ROM folders.

Example:

```
/userdata/.roms_base
/media/roms1/roms
/media/roms2/roms
```

These locations are combined into:

```
/userdata/roms
```

## Graphical Storage Manager File Placement Policy

The graphical Storage Manager currently uses these base MergerFS options:

```
cache.files=partial
dropcacheonclose=true
category.create=eplfs
allow_other
inodecalc=path-hash
moveonenospc=true
minfreespace=4G
```

When the `refresh_pool` path rebuilds an already configured pool, it appends the MergerFS `nonempty` option. The initial merge/rebuild paths use the base option string shown above. This is an implementation detail rather than a user-selectable Storage Manager setting.

**eplfs** means:

```
Existing Path, Least Free Space
```

When writing a new file:

- MergerFS looks for branches where the target folder already exists
- Among those branches, the branch with the least free space is preferred
- Branches with less than 4 GB free are ignored
- If a drive becomes full during a write operation, the file can automatically move to another branch

Example:

```
/media/roms1/roms/nes   900 GB free
/media/roms2/roms/nes   200 GB free
```

A new ROM copied into:

```
/userdata/roms/nes
```

would be stored on:

```
/media/roms2/roms/nes
```

because it has the least free space while still meeting the minimum free space requirement.

The internal `.roms_base` directory is a normal writable branch in the graphical pool. It therefore participates in the `eplfs` policy too. Expanding storage does **not** mean every future write is forced onto an external drive. The destination depends on which branches already contain the target path and their available space.

Because `eplfs` is an existing-path policy, it prefers branches where the relative parent path already exists. Newly formatted Storage Manager drives receive a copy of the standard ROM directory structure specifically so those system directories are available as placement candidates.

The MergerFS implementation and file placement policy may change between Batocera releases. The `eplfs` policy described above applies to the graphical Storage Manager implementation. The manual `S12mergerfs` boot service currently uses `mfs` instead, as described in the Network/NAS section.

### Important MergerFS behavior

MergerFS combines directory trees. It does not stripe or split one ROM file across multiple drives. The free space shown for a pool can therefore be larger than the space available for any one file on an eligible branch. Path-preserving policies, branch modes and `minfreespace` can also produce a no-space error even when the pool appears to have free capacity.

`moveonenospc=true` only helps when a **write to an already created file** fails with a no-space or quota error and MergerFS can move that file to another eligible branch. It does not override every create-policy failure.

Avoid keeping duplicate files at the exact same relative path on multiple branches. MergerFS deduplicates directory entries, and its default search/action policies can select or act on matching paths across branches in ways that are not obvious from EmulationStation.

It is possible to access the underlying branch directories directly, for example under `/media`, while the pool is mounted. However, MergerFS upstream advises against modifying the same file both through a branch path and through the union at the same time, especially when caching is enabled. For the graphical Storage Manager, prefer writing through `/userdata/roms` while the pool is active.

## Troubleshooting and Diagnostics

The main **graphical Storage Manager backend** log is:

```
/var/log/batocera-storage.log
```

The manual `S12mergerfs` init script does not write its own messages to that file. Its `echo` output belongs to the boot/init-service output path, so a clean `batocera-storage.log` does not prove that a manual `premerge` or `S12mergerfs` pool succeeded.

Useful read-only checks from SSH include:

```
batocera-settings-get -f /boot/batocera-boot.conf mergerfs.roms
batocera-storage-manager list_ejectable
batocera-storage-manager list_disks
batocera-storage-manager list_fstypes
findmnt /userdata/roms
mount | grep mergerfs
```

Storage Manager communicates with EmulationStation through its local HTTP API for merge/format prompts, notifications and game-list reloads. The backend waits for EmulationStation to report that it is idle, retrying once per second for up to 300 attempts. A prompt or reload can therefore be delayed while EmulationStation is busy, and after roughly five minutes the request is abandoned and a timeout is written to the Storage Manager log.

For the older `sharedevice=NETWORK` / `sharedevice=DEVICES` mechanism, `S11share` writes diagnostic output to:

```
/tmp/mountDevicesOrNetwork.log
/tmp/mountDevicesOrNetwork.err
```

If a manual NAS-backed `S12mergerfs` pool is not working, first verify that the `premerge` mount actually exists with `findmnt` or `mount` before debugging MergerFS itself.

# Frequently Asked Questions

## Can I use internal drives?

Yes. Any drive mounted by Batocera containing a root-level `roms` folder can be merged into the Storage Manager.

Example:

```
/media/roms1/roms
/media/roms2/roms
```

## Which filesystems can Storage Manager format?

The implementation documented here exposes:

```
ext4
btrfs
exfat
```

Existing drives may use other filesystems if Batocera can mount them. NTFS is explicitly handled for existing drives, but it is not one of the Storage Manager format targets.

### Why does a small blank drive not produce a format prompt?

The current safety logic suppresses the automatic blank-disk format prompt for disks of 4 GiB or smaller. This is intended to avoid accidentally offering small firmware/system LUNs for formatting.

### Can I stop Storage Manager from trying to use a particular drive?

Yes. The backend supports an `ignore.drives` list. Ignoring a drive disables Storage Manager automation for that physical drive while still allowing readable filesystems on it to remain mounted normally.

### How do I remove a drive from the ignore list?

The current backend has an `ignore` action but no separate `unignore` action. Ignored drive identifiers are stored in the `ignore.drives` setting in `/boot/batocera-boot.conf`. Remove the relevant identifier from that setting, then reconnect or re-detect the drive (or reboot) so Storage Manager can evaluate it again.

### What if a configured drive is missing when Batocera boots?

A missing configured branch is skipped from the live pool during refresh, but its saved `mergerfs.roms` entry is not automatically removed. If the drive later returns at the same mount path, Storage Manager can refresh the pool automatically. Explicitly ejecting the drive is different because eject removes its branch from the saved configuration.

### Can I use a network drive or NAS in the same ROM pool?

Yes, but not through the graphical Storage Manager interface.

Advanced users can mount a network share with a `premerge` command in `/boot/batocera-boot.conf` and then include that mounted path in a manual MergerFS pool.

For example:

```
premerge1=mkdir -p /media/NAS && mount.cifs //192.168.1.10/Games /media/NAS
-o guest
mergerfs=/userdata/roms@/media/SSD/roms:/media/NAS/roms=RO
```

See Network/NAS Storage (Advanced) for details and limitations.

### What happens if I unplug a merged drive?

Batocera updates the storage pool and refreshes the game lists.

Games stored only on the disconnected drive will temporarily disappear until the drive is connected again.

## Are BIOS files merged?

Not by the graphical Storage Manager. It only manages the ROM pool at:

```
/userdata/roms
```

BIOS files normally remain stored in:

```
/userdata/bios
```

Advanced users can separately mount or union BIOS storage with `sharedevice=NETWORK`, `sharedevice=DEVICES`, or a manual MergerFS pool, but that is outside the graphical Storage Manager.

## Are saves merged?

Not by the graphical Storage Manager. Save files remain in the normal userdata location, for example:

```
/userdata/saves
```

Advanced `sharedevice` or manual MergerFS configurations can target saves separately, but the Storage Manager UI does not manage that.

## Can I store ROMs in a different folder?

For automatic detection through the graphical Storage Manager, no. It currently expects a root-level folder named:

```
roms
```

For an advanced manual MergerFS pool, yes. The branch can point to an explicitly configured nested directory, for example:

```
/media/SDCARD/path/to/roms
```

## Does the drive name matter?

No.

For example, both of these are valid if they contain a root-level `roms` folder:

```
/media/roms1/roms
```

```
/media/MyGamesDrive/roms
```

## References

- Batocera TV - `Batocera 43 - Episode II`
  - Storage Manager demonstration:
    - https://youtu.be/XpH27Yswe38?t=143
- Batocera Storage Manager source code:
  - https://github.com/batocera-linux/batocera.linux/blob/master/package/batocera/core/batocera-scripts/scripts/batocera-storage-manager
- Batocera manual MergerFS boot service:
  - https://github.com/batocera-linux/batocera.linux/blob/master/package/batocera/utils/mergerfs/S12mergerfs
  - Source snapshot used for the original NAS/MergerFS documentation request:
    - https://github.com/batocera-linux/batocera.linux/blob/7b9aefc9456e5517f0a333bb0caab94938d80c58/package/batocera/utils/mergerfs/S12mergerfs
- Batocera share service (NETWORK / DEVICES):
  - https://github.com/batocera-linux/batocera.linux/blob/master/board/batocera/fsoverlay/etc/init.d/S11share
- Batocera disk formatter:
  - https://github.com/batocera-linux/batocera.linux/blob/master/package/batocera/core/batocera-scripts/scripts/batocera-format
- MergerFS documentation:
  - https://trapexit.github.io/mergerfs/latest/
- Batocera NAS documentation:
  - https://wiki.batocera.org/store_games_on_a_nas
- Batocera external storage documentation:
  - https://wiki.batocera.org/store_games_on_a_second_usb_sata_drive
- Batocera changelog:
  - https://batocera.org/changelog

From:
https://wiki.batocera.org/ - **Batocera.linux - Wiki**

Permanent link:
**https://wiki.batocera.org/user:rion?rev=1791161236**

Last update: **2026/10/05 00:47**