EEOS EE-Branded Software Manual

Version 1.0 – Energy Enhancement Operating System

Complete Reference for All EE Applications

How This Manual Is Organized

Every application in EEOS that carries the EE prefix is documented here. Each entry covers what the app does, how it works internally, every menu item and option, configuration files and paths, command-line usage, integration with other EE components, and detailed troubleshooting. This is the reference that EEAI uses to answer questions about the system, and the reference that field technicians use to diagnose and repair center deployments.

The apps are grouped by function: System Information, Configuration, Software Management, Session and Storage, Installation and Imaging, Utilities, Remote Management, Kiosk and Lockdown, Biometrics, and Boot System.

Part 1: System Information

EESysInfo

Binary: `/usr/local/bin/eeos-sysinfo` **Launcher script:** `/usr/local/bin/eeos-sysinfo` (calls the gtkdialog-based UI at `/usr/local/eeos-sysinfo/`) **Desktop entry:** `eeos-sysinfo.desktop` **Menu location:** System > EESysInfo **Version:** 3.6.4 **License:** GNU GPLv3

Purpose

EESysInfo is a graphical interface providing quick access to more than fifty system utilities that report on hardware, drivers, logs, network, and system configuration. It's the single most useful diagnostic tool for getting a complete picture of what a machine is doing and what it's working with.

Architecture

EESysInfo is built on `gtkdialog`, using XML resource files (`.rc`) to define the window layout, menus, and button actions. Each menu item invokes a shell command or script, captures the output, and displays it in a scrollable text pane. The main executable is a shell wrapper that sets up the environment and launches the gtkdialog UI.

The application directory structure:

```
/usr/local/eeos-sysinfo/
    eeos-sysinfo          # main launcher (called from /usr/local/bin/eeos-sysinfo)
    eeos-sysinfo.rc       # gtkdialog XML defining the full UI
```

```
    Help.html              # built-in help page
    icons/                 # menu icons
```

Menus and Functions

FILE MENU

- **Summary Report:** Generates a concise hardware overview – CPU model, RAM, storage devices, PCI devices, kernel version, boot mode. Suitable for quick triage.
- **Complete Report:** Full hardware dump including everything from the summary plus BIOS tables (DMI decode), all PCI devices with driver info, USB tree, sensor readings, interrupt assignments, DMA channels. This is the report you attach to a support request.
- **Privacy option:** Both reports can strip identifying information (hostname, MAC addresses, serial numbers) before export.
- **Export to text editor:** Sends the current view to the default text editor. Works best with a monospace font to preserve alignment.

MAINBOARD MENU

- **BIOS:** `dmidecode` output showing BIOS vendor, version, date, and capabilities.
- **CPU:** `/proc/cpuinfo` parsed – model name, cores, threads, frequency, cache sizes, flags (SSE, AVX, AES-NI).
- **DMA:** Direct Memory Access channel assignments from `/proc/dma`.
- **DMI:** Full Desktop Management Interface decode – system manufacturer, product name, serial number, chassis type, baseboard info, memory slot details, processor socket info.
- **I/O Resources:** `/proc/ioports` – I/O port address ranges and the devices claiming them.
- **IRQ:** Interrupt request assignments from `/proc/interrupts` – shows which hardware devices are using which interrupt lines and how many interrupts each has processed.

DEVICES MENU

- **Drive Storage:** Submenu with:
  - CD/DVD: optical drive detection and capabilities
  - HDD/SSD: `lsblk`, `hdparm -I` for drive identity, SMART data via `smartctl` if available
  - RAID: software RAID status from `/proc/mdstat`
  - USB Storage: mounted USB drives and their filesystems
  - SCSI: SCSI device tree from `lsscsi`
- **PCI:** Full PCI device tree via `lspci -v`. Submenu for verbose mode, kernel driver info, and numeric IDs.
- **USB:** USB device tree via `lsusb -v`. Shows hierarchy, device descriptors, and claimed drivers.
- **Audio:** ALSA card list, mixer state, EEAudio sink info.
- **Display:** GPU identification, current driver, resolution, refresh rate, OpenGL renderer string.
- **Input:** Keyboard, mouse, touchpad, touchscreen detection.
- **Battery:** Notebook battery health, charge level, design vs full capacity, cycle count (from `/sys/class/power_supply/`).

- **Memory:** Physical memory slots from DMI decode – which slots are populated, DIMM type, speed, manufacturer.
- **Network:** Network interface list, link state, IP addresses, MAC addresses.
- **Printers:** CUPS printer list and status.
- **Sensors:** `sensors` output from lm-sensors if available – CPU temperature, fan speeds, voltages.

DRIVERS MENU

- **Summary:** `lsmod` output – loaded kernel modules with size and dependency count.
- **Complete:** `modinfo` for every loaded module – full description, version, author, firmware requirements, parameters, aliases.
- **EEKView link:** Opens EEKView directly from EESysInfo for deeper module management.

LOGS MENU

- **bootinit.log:** Early boot messages from the init process.
- **bootsysinit.log:** System initialization log from rc.sysinit.
- **Kernel Log:** `dmesg` – the kernel ring buffer. Shows hardware detection, driver loading, errors, and warnings from boot to now.
- **Login Report:** Login/logout activity.
- **System Log:** `/var/log/messages` or journalctl output.
- **xerrs.log:** X server error log (catches X11 protocol errors that don't show in Xorg.0.log).
- **Xorg.0.log:** Full X server log – driver selection, mode setting, monitor detection, input device initialization.

NETWORK MENU

- **Network Config:** Interface configuration, routing table, DNS resolvers.
- **Samba:** Samba server status, share list, connected clients.
- **Firewall:** `iptables -L` or `nftables` ruleset showing active firewall rules.
- **ipinfo:** Scrollable output of the `ipinfo` utility showing public IP, geolocation, ISP.

SYS-APPS MENU

- **System tools:** Links to commonly needed system utilities.
- **Task Manager:** Opens `lxtask` or the configured process manager.
- **Calendar:** Opens `minixcal` or `osmo`.

SYS-FILES MENU

- **Built-in packages:** List of packages that shipped with the EEOS image.
- **Command history:** Shell command history.
- **Installed packages:** Full package database.
- **Session saves:** List of saved sessions and their sizes.
- **Startup files:** rc scripts, autostart entries.
- **SFS layers:** Currently loaded SFS modules and their mount points.

SYS-SPECS MENU

Quick-access system profiles: - **Hardware summary:** One-screen hardware overview (the "bring this to the phone call" report). - **Computer ID:** Unique machine identifier based on hardware serial numbers. - **Distro specs:** EEOS version, build date, kernel version, desktop environment. - **Filesystems:** Mounted filesystems with types, sizes, usage. - **Fonts:** Installed font families. - **Kernel:** Kernel version, build config, command line. - **Locale:** Current language, character set, timezone. - **Menu apps:** All applications registered in the desktop menu. - **Partitions:** Partition table for all detected drives. - **Processes:** Running process list sorted by CPU or memory. - **Session status:** Current session save state, overlay size, persistence mode. - **Environment variables:** Full environment dump.

CLI Usage

The main `eeos-sysinfo` binary is a thin shell script:

```
#!/bin/sh
echo "EEOS System Information"
echo "======================="
# ... outputs CPU, RAM, boot mode to stdout
```

For quick CLI diagnostics without the GUI, this prints a one-screen summary. For the full graphical interface, it must be run from within a desktop session where gtkdialog is available.

Configuration

EESysInfo stores its configuration in: - `/usr/local/eeos-sysinfo/eeos-sysinfo.rc` – gtkdialog layout and menu definitions - Default text editor is read from the system default apps configuration (see EEDefaults)

Integration with Other EE Apps

- Links directly to **EEKView** for kernel module management
- Uses the same hardware detection paths as **EEDiagnostics**
- Reports can be exported and sent via **EERemote** for remote diagnosis
- **EEControl** includes a shortcut to EESysInfo via the sysinfo default app setting

Troubleshooting

Blank reports: The system may be in a minimal boot mode (CLI only or Recovery Shell) where some hardware enumeration tools aren't available. Boot in Standard mode for full hardware access.

Sensor temperatures show "N/A": The hardware doesn't expose thermal data through standard interfaces, or the sensor kernel module isn't loaded. Open EEKView, search for your chipset's sensor module (typically `coretemp` for Intel, `k10temp` for AMD), and load it.

Report text is misaligned: Set the text editor to a monospace font (Noto Mono, DejaVu Sans Mono, or Courier). The reports use fixed-width formatting.

EESysInfo won't open: Check that gtkdialog is installed: `which gtkdialog`. If missing, the system SFS may be corrupted. Also check that the EEOS display server is running (`pgrep labwc`).

DMI/BIOS info shows "To Be Filled By O.E.M.": This is a hardware issue – the manufacturer didn't populate the BIOS DMI tables. Common on cheap motherboards. The hardware still works; you just can't read the serial numbers.

EEKView

Binary: `/usr/local/bin/eeos-kview` (exec wrapper to `/usr/local/eeos-kview/eeos-kview`) **Application directory:** `/usr/local/eeos-kview/` **Desktop entry:** `eeos-kview.desktop` **Menu location:** System > EEKView **Version:** 1.4.1 **License:** GNU GPLv3

Purpose

EEKView is a multi-function viewer for active (loaded) kernel modules, including all hardware drivers. It shows what drivers the kernel has loaded, what hardware they're attached to, their memory footprint, version, dependencies, and whether they're built-in or dynamically loaded.

Architecture

Like EESysInfo, EEKView is a gtkdialog application. The main UI is split into two panes: - **Top pane:** Scrollable list of all loaded kernel modules with brief descriptions - **Bottom pane:** Detailed information for the selected module (concatenated output of `lsmod` and `modinfo`)

Application directory:

```
/usr/local/eeos-kview/
    eeos-kview            # main launcher
    Help.html             # built-in help
```

Menus and Functions

FILE MENU

- **Refresh:** Re-reads the module list. Use after manually loading or removing a module.
- **Export:** Sends the current module detail view to the default text editor.

MANAGER MENU

- **Black List:** Select a problematic module to prevent it from loading on next boot. Writes to `/etc/modprobe.d/blacklist-eeos.conf`.
- **Load Module:** Load an inactive (available but not loaded) module by name. Uses `modprobe`.
- **Competing Modules:** When two drivers claim the same hardware (e.g., `nouveau` vs `nvidia`, `r8169` vs `r8168`), this lets you choose which one gets priority. Writes to the EEOS module preferences list in `/etc/rc.d/MODULESCONFIG`.

LOGS MENU

- **Kernel messages (dmesg):** Full kernel ring buffer. Filter for driver-related messages by searching for module names.
- **System activity (/var/log/messages):** System daemon messages including module load/unload events.

REPORTS MENU

- **Summary (lsmod):** Tabular view of all loaded modules with size and use count.
- **Complete (modinfo):** Full `modinfo` output for every loaded module – description, version, author, license, firmware files, parameters, aliases, and the .ko file path.
- **Firmware report:** Lists all modules that depend on firmware files, whether loaded or not. Shows which firmware files are present in `/lib/firmware/` and which are missing.
- **PCI report:** PCI device list with the kernel driver bound to each device.
- **USB report:** USB device list with driver binding info.

Module Selection

Click any module in the top list. The bottom pane shows: - **Module name** and description - **Size** in bytes - **Used by** count and list of dependent modules - **File path** (e.g., `/lib/modules/6.x.x/kernel/drivers/net/e1000e/e1000e.ko`) - **Version** string - **Parameters** the module accepts (e.g., `debug=0`, `speed=auto`) - **Firmware** files the module requests at load time - **Aliases** (modalias strings that trigger auto-loading)

Integration

- **EESysInfo** links to EEKView from its Drivers menu
- **EEEvents** backend (`eeos-event-backend-modprobe`) uses the same module preference system that EEKView's Manager menu configures
- Module blacklisting affects the **EEAI Boot Profile Selector** since blacklisted modules change hardware detection behavior

Troubleshooting

Device not working: Open EEKView, search (Ctrl+F in the text view) for the device vendor name or chip name. If no module is loaded for it: 1. Check if a module exists: run `modinfo <module_name>` from a terminal 2. Try loading it: Manager > Load Module 3. If it loads but doesn't stick across reboots, add it to `/etc/modules-load.d/eeos.conf`

Accidentally blacklisted a critical module (e.g., network driver): 1. Boot using "Recovery Shell" from the GRUB menu 2. Mount the save partition if using persistence 3. Edit `/etc/modprobe.d/blacklist-eeos.conf` and remove the offending line 4. Reboot

Two drivers fighting over the same hardware: Symptoms: device works intermittently, dmesg shows repeated bind/unbind messages. Fix: Manager > Competing Modules, select the preferred driver. This writes a preference entry that the EEOS event backend respects during hardware detection.

Module loads but device still doesn't work: Check dmesg for firmware load failures: `dmesg | grep -i firmware`. If firmware files are missing, they need to be added to `/lib/firmware/`. The `eeos_firmware.sfs` module should contain most common firmware, but some proprietary firmware (like certain WiFi chips) may need manual installation.

EEDiagnostics

Binary: `/usr/local/bin/eeai-diag` **Desktop entry:** `eeai-diag.desktop` **Menu location:** System > EEDiagnostics **Version:** 1.0.0

Purpose

Runtime health checker for EEOS systems. Runs a comprehensive check of CPU, RAM, thermals, boot mode, HHFE/DOSBox status, storage persistence, LUKS encryption state, network connectivity, Tailscale VPN, and USB devices. Generates a diagnostic report file and offers interactive repair actions for common problems.

Architecture

Pure bash script with no GUI dependencies – works in both desktop and CLI environments. Uses ANSI 256-color escapes for the EEOS visual theme (teal/gold/sage palette matching eesystem.com design language). Falls back to plain text when not running in a terminal or when `--no-color` is passed.

The color palette: - Teal (#68c4b2, xterm-256 slot 116): accents and decorative elements - Gold (#f0b55a, xterm-256 slot 215): strong accents and repair menu numbers - Sage (#d4e5e2, xterm-256 slot 188): body text - Cream (#f5f0d5, xterm-256 slot 230): section headers - Green (slot 79): healthy/OK states - Amber (slot 214): warnings - Red (slot 167): errors

Interactive Menu

When launched without arguments, EEDiagnostics displays the EEOS ASCII banner and an interactive menu:

```
1  System Status
2  HHFE Checks
3  Storage & Persistence
4  Network
5  Thermal & Hardware
6  Repair Actions
7  Generate Full Report
8  Exit
```

Commands (CLI and Interactive)

SYSTEM STATUS (`EEAI-DIAG STATUS`)

Checks: - **CPU:** Model name, core count, current frequency in MHz - **RAM:** Total, used, available, percentage. Color-coded: - Green: under 80% used - Amber: 80-95% used - Red: over 95% used (critical) - **Swap:** Reports if swap is active (unusual for EEOS since it runs from RAM or overlay) - **Boot mode:** Reads `/proc/cmdline` for EEOS boot flags: - `eeos=ram` → RAM-only (ephemeral) - `eeos=copy` → Copy-to-RAM - `eeos=nox` → CLI only - `eeos=rdsh` → Ram Disk Shell - `eeai` → EEAI Assistant mode - default → Standard - **Uptime:** Human-readable format (e.g., "2d 5h 23m") - **Root filesystem type:** overlay, tmpfs, ext4, etc. - **Overlay layers:** Count of SFS layers in the overlay stack

HHFE CHECKS (`EEAI-DIAG HHFE`)

Checks: - **DOSBox process:** Searches for running `dosbox` process via `pgrep` - **Resource usage:** CPU%, memory%, elapsed time for the DOSBox process - **INVOKE.TXT:** Verifies the HHFE invocation file exists at `/`

`opt/hhfe/INVOKE.TXT` (or encrypted partition override at `/mnt/data/hhfe/INVOKE.TXT`) - **DOSBox config:** Looks for `dosbox-hhfe.conf` at `/opt/hhfe/.dosbox/dosbox-hhfe.conf` and alternate locations

If DOSBox is not running, offers to restart it automatically.

STORAGE & PERSISTENCE (`EEAI-DIAG STORAGE`)

Checks: - **Boot device:** Identifies the device EEOS booted from (USB, HDD, etc.) by checking `/proc/mounts` for live/boot mount points - **Media type:** Checks `/sys/block/<dev>/removable` to determine if the boot device is USB - **LUKS encryption:** - If `eeos-data` mapper device exists: reports active encrypted volume and mount point - If a `crypto_LUKS` partition exists but isn't unlocked: warns - If no LUKS partition: reports as not detected - **Persistence mode:** Detects from mount table and kernel command line: - `eeossave.4fs` → savefile mode - `eeossave` folder → savefolder mode - `eeos=ram` → RAM-only (no persistence) - **Filesystem usage:** `df -h` for non-virtual filesystems, color-coded by usage (red at 90%+, amber at 75%+)

NETWORK (`EEAI-DIAG NETWORK`)

Checks: - **Interfaces:** `ip -br addr show` listing all network interfaces with state and addresses - **Default gateway:** From routing table - **DNS servers:** From `/etc/resolv.conf` - **Gateway reachability:** ICMP ping to gateway (2-second timeout) - **External connectivity:** Ping to 8.8.8.8 (3-second timeout) - **EENetwork state:** Reports the network service state (online, ready, idle, etc.) - **Tailscale:** If installed, reports Tailscale status and self-identity

THERMAL & HARDWARE (`EEAI-DIAG THERMAL`)

Checks: - **CPU temperature:** Reads all thermal zones from `/sys/class/thermal/thermal_zone*/temp` - Green: under 75°C - Amber: 75-90°C (elevated) - Red: over 90°C (critical) - **Fan speeds:** Reads from `/sys/class/hwmon/hwmon*/fan*_input` when available - **USB devices:** `lsusb` output (or fallback to sysfs product names) - **Key PCI devices:** `lspci` filtered for VGA, audio, network, ethernet, wifi, and bridge devices

REPAIR ACTIONS (`EEAI-DIAG REPAIR`)

Interactive submenu with confirmation prompts. Nothing runs without explicit "y" confirmation.

1. **Restart HHFE (DOSBox):**
   - Kills existing DOSBox process
   - Waits 1 second for clean shutdown
   - Tries `/opt/hhfe/hhfe-start.sh` first
   - Falls back to reading `Exec=` from `/etc/xdg/autostart/hhfe-autostart.desktop`
   - Waits 2 seconds, verifies DOSBox is running
   - Reports success or failure
2. **Restart EENetwork service:**
   - Uses `systemctl restart connman` if systemd is available
   - Falls back to killing and restarting the network daemon directly
   - Waits 2 seconds for network to come back
3. **Restart EEDesktop compositor:**
   - Kills the compositor process (screen will flash/go black momentarily)
   - Waits 2 seconds for session manager to restart it

- If the compositor doesn't restart automatically, launches it manually

4. **Remount persistence layer:**
   - Finds the `eeossave` mount from `/proc/mounts`
   - Runs `mount -o remount` on it
   - Useful when the overlay has gotten into a weird state and changes aren't writing
5. **Clear tmpfs caches:**
   - Syncs filesystem buffers
   - Writes `3` to `/proc/sys/vm/drop_caches` to drop page cache, dentries, and inodes
   - Requires root

GENERATE FULL REPORT (`EEAI-DIAG REPORT`)

Creates a timestamped plain-text report at `/tmp/eeos-diagnostic-report-YYYYMMDD-HHMMSS.txt` containing: - All five diagnostic sections (status, HHFE, storage, network, thermal) - Full mount table - Top 30 processes by CPU usage - Last 50 lines of kernel messages (dmesg) - Kernel command line

Color codes are stripped for the report file so it's clean text suitable for email or USB export.

Command-Line Interface

```
eeai-diag                    # interactive menu
eeai-diag status             # system overview
eeai-diag hhfe               # HHFE/DOSBox checks
eeai-diag storage            # storage and persistence
eeai-diag network            # network diagnostics
eeai-diag thermal            # hardware and thermal
eeai-diag repair             # interactive repair menu
eeai-diag report             # generate report file
eeai-diag --no-color         # disable color output
eeai-diag --version          # show version
eeai-diag --help             # usage info
```

Troubleshooting the Diagnostics Tool Itself

Colors look wrong in terminal: Use `--no-color` flag, or ensure your terminal supports 256-color mode. The default `lxterminal` and `urxvt` in EEOS both support it.

"Need root to clear caches" warning: Cache clearing requires root privileges. Run as root or use `sudo eeai-diag repair`.

Report file not created: Check that `/tmp` has free space: `df -h /tmp`. In RAM-only mode, `/tmp` is a tmpfs that shares RAM with the system.

EEEvents

Binary: `/usr/local/bin/eeos-event-manager` (simple log viewer)
Frontend: `/usr/local/eeos-event/eeos-event-frontend` (desktop daemon) **Backends:** `/usr/sbin/eeos-event-backend-firmware`, `/usr/sbin/eeos-event-backend-modprobe` **Autostart:** `fixmenusd.desktop` (via xinitrc) **Desktop entry:** `eeos-event-manager.desktop` **Menu location:** System > EEEvents

Purpose

The EEOS event system is a three-part architecture that handles hardware events at the kernel level:

1. **Event backends** (called by udev rules): Handle firmware loading requests and kernel module auto-detection
2. **Event frontend** (desktop daemon): Displays hardware events to the user via desktop notifications
3. **Event manager** (log viewer): Shows recent system events for debugging

Event Backend: Firmware (`eeos-event-backend-firmware`)

Called by udev rule `/etc/udev/rules.d/50-udev-eeos-basic.rules` when the kernel requests firmware for a device.

Process: 1. Extracts the firmware filename from the `$FIRMWARE` environment variable 2. Searches `/lib/firmware/` recursively for the file 3. If found: writes `1` to `/sys$DEVPATH/loading`, copies the firmware data to `/sys$DEVPATH/data`, writes `0` to `/sys$DEVPATH/loading` 4. If not found: writes `-1` to `/sys$DEVPATH/loading` (tells the kernel the firmware isn't available)

This is critical for WiFi adapters, Bluetooth controllers, and GPU firmware that require userspace firmware loading.

Event Backend: Module Probe (`eeos-event-backend-modprobe`)

Called by udev when a new hardware device is detected (hotplug or at boot). This is the core of EEOS's hardware auto-detection.

Process: 1. Reads `$MODALIAS` from the udev environment 2. Runs `modprobe --show-depends $MODALIAS` to determine which kernel module handles this device 3. Checks the EEOS preference list (`/etc/rc.d/MODULESCONFIG`) for competing module overrides 4. For PCI devices: checks vendor/device ID against the PCI_OVERRIDES table for manual driver assignments 5. Uses a file-based locking mechanism (`/tmp/eeos_event_backend/`) to prevent race conditions when multiple devices are detected simultaneously 6. Loads the module via `modprobe`

The preference system is what EEKView's "Competing Modules" manager configures. Format in MODULESCONFIG:

```
PREFLIST="module_a:module_b:module_c"
```

Where the last entry is most preferred.

Event Frontend (`eeos-event-frontend`)

Desktop daemon started from `.xinitrc`. Monitors udev events and displays desktop notifications when: - A USB device is plugged in or removed - A network interface comes up or goes down - A display is connected or disconnected - A firmware load succeeds or fails - A kernel module is loaded or fails to load

Runs as root (required for udev access). Only one instance runs at a time (checked via process table).

Debug mode: `eeos-event-frontend -debug` writes to `/tmp/eeos-event-frontend.log`.

Event Manager (Log Viewer)

Simple CLI tool that displays recent system events:

```
#!/bin/sh
echo "EEOS Event Log"
if command -v journalctl >/dev/null 2>&1; then
    journalctl --no-pager -n 50
else
    tail -50 /var/log/messages || tail -50 /var/log/syslog || dmesg | tail -50
fi
```

Troubleshooting

USB device plugged in but nothing happens: 1. Check EEEvents log – did the kernel detect the device? (`dmesg | tail -20`) 2. If the device appears in dmesg but no module loaded, the backend couldn't find a matching module. Check `modinfo` for the device's modalias. 3. If firmware loading failed, check that `eeos_firmware.sfs` is loaded: `mount | grep firmware`

Module preferences not being respected: Check `/etc/rc.d/MODULESCONFIG` for the PREFLIST entry. The format uses colons as separators with the preferred module last. After changing preferences via EEKView, a reboot is required for the change to take effect on boot-time hardware detection.

Part 2: Configuration

EEControl

Binary: `/usr/local/bin/eeos-control` (exec wrapper) **Application directory:** `/usr/local/eeos-control/` **Config file:** `/usr/local/eeos-control/eeos-control.rc` **Functions library:** `/usr/local/eeos-control/func` **Desktop entry:** `eeos-control.desktop` **Menu location:** Setup > EEControl **Version:** 3.6

Purpose

Central control panel for EEOS. Provides one-window access to system configuration, default applications, hardware management, power management, and quick-launch tools.

Architecture

EEControl is a gtkdialog application with a modular design. The main window is defined in the `.rc` file, while the `func` script handles all the actual operations as a case statement with dozens of action handlers.

Configuration file (`eeos-control.rc`) defines default application mappings:

```
BACKUP="grsync"
BURNER="pburn"
CONSOLE="defaultterminal"
DOWNLOAD="uget-gtk"
FILE_ARCHIVER="defaultarchiver"
FILE_FINDER="pfind"
FILE_MANAGER="defaultfilemanager"
FTP="gftp-gtk"
GRUB="grub2config"
NETWORK="connectwizard"
NOTEBOOK="cherrytree"
ORGANIZER="osmo-wrapper"
P2P="transmission-gtk"
SAMBA="samba.sh"
SCANNER="simple-scan"
SYSINFO="hardinfo2"
TASK_MANAGER="defaultprocessmanager"
WALLPAPER="qwallpaper"
```

Each entry can be customized. Up to 8 custom control entries (`MYCONTROL1` through `MYCONTROL8`) can be added for center-specific tools.

Functions (from the `func` script)

EEControl's func script accepts command-line arguments to perform specific actions:

- **`-calculator`:** Opens galculator, gcalctool, calcoo, cgtkcalc, or xcalc (first found)
- **`-cpufrequency`:** Opens CPU frequency scaling control (wcpufreq, cpu_freq, or acpi-control)
- **`-firewall`:** Opens the firewall manager (firewall_ng or Firewall_Genie)
- **`-fonts`:** Font size adjustment dialog with three options:
  - Taskbar: opens EETaskbar configuration
  - DPI Scale: opens wdisplays (Wayland display settings)
  - Desktop: opens EEFiles preferences (file manager font settings)
- **`-gparted`:** Opens GParted partition editor
- **`-gtkhash`:** Opens hash/checksum verification tool
- **`-hwclock`:** Hardware clock configuration (UTC vs local time)
- **`-inxi`:** Runs `inxi -Fxz` in a scrollable dialog – quick system info dump with privacy flag
- **`-logout`:** Logs out of the desktop session

- `-menu_update:` Forces a rebuild of the XDG desktop menu (runs `fixmenus`)
- `-mount:` Opens the partition mount manager (pmount or ymount)
- `-poweroff:` Shuts down the system
- `-preferences_save:` Saves current control panel preferences
- `-preferences_nochange:` Reverts preferences to last saved state
- `-printmanager:` Opens print job manager (gtklpq/gtklp) or CUPS web interface
- `-printsetup:` Opens CUPS printer setup shell
- `-reboot:` Reboots the system
- `-updatetime:` Syncs system clock (qsync, psync)
- `-wallpaper:` Opens wallpaper chooser (qwallpaper)
- `-webcam:` Opens webcam viewer (guvcview, luvcview, or ucview)
- `-eeos-display-wizard:` Opens EEDisplay

Preferences Panel

EEControl has a built-in preferences editor that lets you change which applications are launched for each category (backup, burner, console, etc.). Changes are written to `eeos-control.rc` and `eeos-control2.rc` (backup copy).

Troubleshooting

EEControl won't open: Check that gtkdialog is installed and a desktop session is running. Try from terminal: `/usr/local/eeos-control/eeos-control`.

A button does nothing: The application it's trying to launch isn't installed. EEControl checks `which <app>` before launching and shows a "not available" dialog if the app is missing. Install the missing application via EEPackages or EESoftware.

Custom entries not showing: Edit `/usr/local/eeos-control/eeos-control.rc` directly and set `MYCONTROL1` through `MYCONTROL8` to the desired commands.

EESetup

The EEOS Setup Wizard. Handles locale, timezone, keyboard layout, screen resolution, and network configuration. Runs automatically on first boot to get the machine usable fast, and stays available from the menu for any time you need to change regional or display settings afterward.

Binary: `/usr/sbin/eeos-setup` **Symlinks:** `chooselocale`, `timezone-set`, `chooselayout`, `countrywizard`, `quickcountry`, `xrandrshell` (all symlink to `eeos-setup`) **Desktop entry:** `eeos-setup.desktop` **Menu location:** Setup > EESetup **Icon:** `gtk-preferences` **Autorun:** First boot (via `welcome1stboot`)

How It Works

EESetup is a single script that changes behavior based on how it's invoked. The main binary lives at `/usr/sbin/eeos-setup`, and several symlinks point to it. When you call it by a different name, it opens only the relevant panel instead of the full wizard.

Invocation name	What opens
`eeos-setup`	Full wizard – all panels
`chooselocale`	Language/locale panel only
`timezone-set`	Timezone panel only
`chooselayout`	Keyboard layout panel only
`eeos-countrywizard`	Locale + timezone + keyboard bundled
`eeos-quickcountry`	Same as above
`xrandrshell`	Screen resolution panel only

This means other EEOS tools can call specific panels without pulling up the entire wizard. EESetup Hub, for instance, links to individual panels through these symlink names.

First Boot Behavior

On a fresh EEOS deployment, the init system calls `eeos-setup` as part of `welcome1stboot`. Here's what happens:

1. A welcome splash appears with the EEOS logo (`eeos.svg`) and a "Welcome to EEOS!" greeting.
2. The full wizard opens with all panels visible – locale, timezone, keyboard, resolution, and network.
3. The user sets their preferences and clicks OK.
4. EESetup applies everything at once: generates locale files, sets the timezone symlink, writes the keyboard config, applies the screen resolution, and optionally configures the network.
5. `fixdesk` and `fixmenus` run to translate desktop entries and menus to the selected language.
6. If the system detected an ethernet connection during boot, the hostname was already auto-configured.

The welcome splash can be suppressed with `eeos-setup nosplash` if you're scripting a deployment and don't want the GUI greeting.

The Panels

LOCALE (LANGUAGE) PANEL

Sets the system language and regional formatting.

What you see: A dropdown listing every available locale in `language_COUNTRY.encoding` format (e.g., `en_US.UTF-8`, `de_DE.UTF-8`, `pt_BR.UTF-8`).

What happens when you change it:

1. `localedef` runs to generate the locale data files if they don't already exist.
2. The selection is written to `/etc/locale`.
3. `fixdesk` and `fixmenus` run to retranslate all desktop entries and menu files to the new language.
4. If the selected language needs a language pack that isn't installed, EESetup offers to download it.
5. Optionally, EESetup can patch the initrd to display the new language during early boot messages.

The language-change popup: When you switch languages, the confirmation dialog itself is translated. Hardcoded translations exist for Dutch, Portuguese, Spanish, Italian, Polish, German, and French. For other languages, it falls back to English.

Config files modified: - `/etc/locale` – the active locale string

Built-in help text (accessible via the ? button):

> The locale setting controls the language used throughout the system – menus, dialog boxes, application interfaces, date and time formatting, number formatting, and currency symbols.
>
> Select your locale from the dropdown. If you don't see your exact locale, pick the closest match for your language and country.
>
> After changing the locale, desktop menus and application names will update to the new language. Some applications may need to be restarted to pick up the change.
>
> If you need a language pack that isn't installed, EESetup will offer to download it. This requires an internet connection.
>
> The initrd translation option updates early boot messages (the text you see before the desktop loads) to match your language. This is optional and can be done later.

TIMEZONE PANEL

Sets the system clock to the correct timezone.

What you see: Two-step selection. First pick a continent or region (Africa, America, Antarctica, Asia, Atlantic, Australia, Europe, Indian, Pacific). Then pick the specific city from that region.

UTC vs Local clock: A separate option lets you choose whether the hardware clock (BIOS/UEFI clock) is set to UTC or local time.

- **UTC (recommended):** The hardware clock stores UTC, and the OS converts to local time. Daylight Saving Time adjustments happen automatically.
- **Local:** The hardware clock stores local time directly. Needed on dual-boot systems where the other OS expects local time (common with Windows).

What happens when you change it:

1. `/etc/localtime` is symlinked to the matching file under `/usr/share/zoneinfo/` (e.g., `/usr/share/zoneinfo/America/New_York`).

2. The UTC/local preference is written to `/etc/clock`.

First-boot note: If `/etc/localtime` doesn't exist when EESetup runs (fresh install), the timezone panel is flagged as mandatory – you can't skip it.

Config files modified: - `/etc/localtime` – symlink to the zoneinfo file - `/etc/clock` – UTC or LOCAL hardware clock setting

Built-in help text:

> Pick your timezone by selecting the continent or ocean first, then the city closest to you. You don't need to be in that exact city – the timezone database groups regions that share the same time rules.
>
> **About DST:** When you select a city-based timezone (like America/New_York), Daylight Saving Time adjustments happen automatically. You never need to change your timezone for DST.
>
> **About GMT entries:** The GMT+N and GMT-N entries in the timezone list are fixed offsets. They don't adjust for DST. Only use these if your region doesn't observe DST and you know your exact offset.
>
> **UTC vs Local clock:** If this machine only runs EEOS, choose UTC. If it dual-boots with another operating system that sets the hardware clock to local time, choose Local to avoid the clock jumping by several hours every time you switch OS.
>
> Use Page Down and Page Up to scroll through long timezone lists.

KEYBOARD LAYOUT PANEL

Sets the keyboard layout for both the console (text mode) and the graphical desktop.

What you see: A dropdown of available console keymaps, sourced from `/lib/keymaps/`. Below that, a numlock on/off toggle.

How the X11 translation works: Console keymaps and X11 keyboard layouts use different naming. When you pick a console keymap, EESetup auto-translates it to the equivalent X11 layout by looking it up in `/etc/X11/xkb/symbols/`. If no exact match is found, it falls back to `us` (US English).

What happens when you change it:

1. The selected keymap name is written to `/etc/keymap`.
2. If a graphical session is running, the X11 layout is applied immediately – no logout needed.
3. The keymap is inserted as `/EEOSKEYMAP` into the initrd for use during early boot (console mode).
4. The numlock preference is saved to `~/Startup/numlockx`. On next login, numlockx reads this and sets numlock accordingly.

Console vs graphical: Keyboard changes take effect immediately in the graphical desktop (X11/Wayland). Console mode (text terminals, recovery shell) uses a different keymap system and requires a reboot to pick up changes.

Config files modified: - `/etc/keymap` – console keymap name - `/EEOSKEYMAP` (in initrd) – early-boot console keymap - `~/Startup/numlockx` – numlock preference

Built-in help text:

> Select the keyboard layout that matches your physical keyboard. The most common layouts are near the top of the list.
>
> **Console vs Desktop:** The layout you select here applies to both the graphical desktop and the text console. However, the console keymap is baked into the boot image (initrd), so console changes require a reboot to take full effect. The graphical desktop updates immediately.
>
> **Numlock:** The numlock toggle controls whether the numeric keypad starts in number mode or navigation mode on login. This is a personal preference – center deployments typically leave it off since most HHFE operation doesn't use the numpad.
>
> **If your layout isn't listed:** The list shows console keymaps. Some less common layouts may not have a console equivalent. In that case, use the closest match here and configure the exact X11 layout through EEControl > Input.
>
> **Console font loading:** The keyboard layout can also affect which console font is loaded during early boot, particularly for non-Latin scripts. If you see garbled text in the console after changing layouts, the console font may need updating through EEControl.

SCREEN RESOLUTION PANEL

Sets the display resolution and refresh rate.

What you see: A list of all resolutions detected by `xrandr`, including the available refresh rates for each.

Live preview: Selecting a resolution applies it immediately so you can see how it looks. If the display goes blank, garbled, or off-screen, EESetup reverts to the previous resolution automatically.

What happens when you confirm:

1. The selected resolution and refresh rate are written to `/etc/X11/xorg.conf`.
2. If the display wizard is needed for multi-monitor configuration, EESetup provides a link to open it.

Raspberry Pi note: On Raspberry Pi hardware, this panel is replaced with audio and video configuration options specific to the Pi's firmware-level display handling, rather than xrandr-based resolution selection.

Config files modified: - `/etc/X11/xorg.conf` – display resolution and refresh rate

Built-in help text:

> Pick the resolution that looks best on your monitor. The list shows every mode your display hardware reported as supported.
>
> The resolution is applied as soon as you select it, so you can see the result before committing. If the screen goes blank or distorted, wait 15 seconds and it will revert automatically.
>
> If you need to configure multiple monitors, use the display wizard (link at the bottom of this panel) or open EEDisplay from the Setup menu.

> If your preferred resolution isn't listed, it means xrandr didn't detect it as a supported mode for your display hardware. You can force custom resolutions through EEDisplay's advanced mode, but this risks damaging some monitors – only do it if you know the monitor's specifications.

NETWORK PANEL

Configures network connectivity, firewall, and time sync.

What you see: A button to open the connection wizard, plus two checkboxes:

- **Firewall:** Enable or disable the EEOS firewall. Enabled by default on center deployments.
- **NTP time sync:** Enable or disable automatic time synchronization over the network. When enabled, the system clock stays accurate by syncing with internet time servers.

First-boot behavior: If the system detected an active ethernet connection during boot, the hostname was auto-configured and network is already working. This panel is mainly for WiFi setup or firewall/NTP preferences.

CLI and Headless Modes

EESetup has three non-GUI invocation modes for scripting and headless deployments:

Mode	Command	When to use
CLI mode	`eeos-setup cli`	Text-mode interface, no graphical display needed. For SSH sessions or console-only deployments.
Compose-only	`eeos-setup composeonly`	Runs configuration logic without a display server. For automated builds and image preparation where you set locale/timezone/keyboard programmatically.
No splash	`eeos-setup nosplash`	Normal GUI, but skips the welcome splash screen. For re-running the wizard after first boot when you don't need the greeting.

GUI Behavior

- **Auto-scaling:** EESetup detects the current font size and scales the GUI accordingly. On high-DPI displays, widgets and text scale up.
- **Window manager detection:** The layout adjusts for different window managers. The behavior is consistent regardless of the compositor in use.
- **Internationalization:** The entire GUI supports gettext-based translation. If the system locale is set to a supported language, all labels, buttons, and help text appear in that language.

- **Frame layout:** The wizard organizes panels into three frames:
  - **Country frame:** Locale, timezone, keyboard
  - **Graphics frame:** Screen resolution, display wizard link
  - **Network frame:** Connection wizard, firewall, NTP

Config Files Reference

Every file EESetup reads or writes:

File	What it controls
`/etc/locale`	Active locale string (e.g., `en_US.UTF-8`)
`/etc/localtime`	Symlink to active timezone in `/usr/share/zoneinfo/`
`/etc/clock`	Hardware clock mode: `UTC` or `LOCAL`
`/etc/keymap`	Active console keymap name
`/etc/X11/xorg.conf`	Display resolution and refresh rate
`/etc/X11/xkb/symbols/`	X11 keyboard layout definitions (read-only lookup)
`/lib/keymaps/`	Available console keymaps (read-only lookup)
`/usr/share/zoneinfo/`	Timezone database (read-only lookup)
`~/Startup/numlockx`	Numlock on/off preference
`/EEOSKEYMAP` (in initrd)	Console keymap for early boot

Troubleshooting

EESetup won't open: - Check that `/usr/sbin/eeos-setup` exists and is executable. - If it opens blank, the GTK dialog engine may not be running. Try from a terminal: `eeos-setup` and check for error output.

Locale change didn't take effect: - Log out and back in, or reboot. Some applications cache the locale at startup. - Check `/etc/locale` to confirm the new value was actually written. - If desktop menus are still in the old language, run `fixmenus` manually from a terminal.

Timezone is wrong after setting it: - Verify `/etc/localtime` points to the correct zoneinfo file: `ls -la /etc/localtime`. - Check `/etc/clock` – if this says LOCAL but the hardware clock is UTC (or vice versa), the time will be off by your UTC offset. - If dual-booting with Windows and the time jumps every reboot, set `/etc/clock` to LOCAL.

Keyboard layout not working in the console: - Console changes require a reboot. The graphical desktop updates instantly, but the console uses the initrd-embedded keymap. - After reboot, if the console layout is still wrong, check that `/etc/keymap` contains the expected value and that the initrd was rebuilt.

Screen resolution reverts or goes blank: - If a selected resolution causes a blank screen, wait 15 seconds – EESetup auto-reverts. - If the auto-revert doesn't happen (hard blank), reboot. EEOS falls back to the last

working configuration. - If no resolutions are listed, xrandr can't talk to the display. Try booting with "Safe Video (nomodeset)" from the boot menu, then re-run EESetup.

Network panel does nothing: - The connection wizard is a separate tool. If clicking it does nothing, the network manager service may not be running. Check from a terminal: `connmanctl state` or equivalent.

First-boot wizard didn't appear: - The init system calls `welcome1stboot`, which calls `eeos-setup`. If you booted into CLI mode or recovery shell, the wizard is skipped. - Re-run manually: `eeos-setup` from a graphical terminal, or `eeos-setup cli` from a text console.

Language pack download failed: - Requires an internet connection. Configure network first (either through the network panel or by plugging in ethernet), then re-run the locale change.

EESetup Hub

Binary: `/usr/sbin/eeos-setup-hub` (exec wrapper to `/usr/local/eeos-setup-hub/eeos-setup-hub`)
Desktop entry: `eeos-setup-hub.desktop` **Menu location:** Setup > EESetup Hub

Purpose

All-in-one configuration dashboard. While EESetup is the quick first-boot wizard (focused on getting things working), EESetup Hub is the full settings panel for changing anything at any time.

Provides access to: - Country, language, timezone, keyboard (links to EESetup subsystems) - Display configuration (links to EEDisplay) - Network management (links to connection wizard) - Firewall configuration - User management - And all other system settings in a categorized interface

When to Use EESetup Hub vs EESetup

- **EESetup:** Run this first time, or when you need to quickly reconfigure the basics (locale, keyboard, resolution, network). Takes 30 seconds.
- **EESetup Hub:** Run this when you need to change something specific after the initial setup is done. More options, more detail, less hand-holding.

EEDisplay

Binary: `/usr/local/bin/eeos-display-wizard` (CLI helper) **Full GUI:** `/usr/sbin/eeos-display-wizard` (the real display wizard, 1000+ lines) **Automatic mode:** `/usr/sbin/eeos-display-wizard-automatic` **CLI mode:** `/usr/sbin/eeos-display-wizard-cli` **Desktop entry:** `eeos-display-wizard.desktop` **Menu location:** Setup > EEDisplay

Purpose

Monitor and display configuration for EEOS desktop sessions. Handles resolution, refresh rate, multi-monitor layout, video driver selection, and color depth.

Architecture

Three modes of operation:

1. **CLI helper** (`/usr/local/bin/eeos-display-wizard`): Simple wrapper that detects Wayland vs X11 and calls `wlr-randr` or `xrandr` accordingly. Commands: `list`, `set <output> <mode>`, or no args for quick status.
2. **Full GUI** (`/usr/sbin/eeos-display-wizard`): The real display wizard. A 1000+ line bash/gtkdialog application that:
   - Detects available display outputs
   - Lists all supported resolutions and refresh rates
   - Provides a preview-and-revert mechanism (if the new resolution makes the screen unreadable, it reverts after a timeout)
   - Configures the Xorg video driver (`/etc/X11/xorg.conf`)
   - Sets color depth (16-bit or 24-bit)
   - Handles modeset/nomodeset for problematic GPUs
3. **Automatic mode** (`eeos-display-wizard-automatic`): Runs at boot without user interaction, auto-detects the best resolution for the connected display.

Key Functions

`SET_XORG_VIDEO_DRIVER()`

Writes the video driver selection to `/etc/X11/xorg.conf`. Options: - `auto`: Comments out the driver line, lets Xorg auto-detect - Specific driver name (e.g., `intel`, `nouveau`, `amdgpu`, `vesa`): Forces that driver

`SET_XORG_COLOR_DEPTH()`

Sets the color depth in xorg.conf: - `16`: 16-bit color (65K colors, lower memory, faster) - `24`: 24-bit color (16M colors, standard) - `remove`: Comments out the depth line (auto-detect)

`SET_XORG_SCREEN_RES()`

Forces a specific resolution in xorg.conf. Format: `1920x1080`, `1280x720`, etc.

Wayland vs X11

On the EEOS Wayland desktop, the display wizard uses `wlr-randr` for output management. On legacy X11 setups, it uses `xrandr` and writes to `/etc/X11/xorg.conf`.

The `/usr/local/bin/eeos-display-wizard` CLI helper auto-detects which session type is running:

```
if command -v wlr-randr >/dev/null 2>&1; then
    # wayland path
elif command -v xrandr >/dev/null 2>&1; then
    # X11 fallback
fi
```

Troubleshooting

Screen blank or distorted after changing resolution: Reboot. EEOS returns to the last known working configuration if the xorg.conf resolution fails. From the GRUB menu, choose "Safe Video (nomodeset)" for a guaranteed working display.

Projector or external monitor not detected: 1. Try the "Force display setup" boot entry 2. From a terminal: `wlr-randr` (Wayland) or `xrandr` (X11) to see detected outputs 3. Some projectors need `nomodeset` to be detected properly

Multi-monitor not working: Under Wayland: use `wdisplays` (graphical tool) or `wlr-randr --output <name> --pos <x>,<y>` from terminal. Under X11: use `xrandr --output <name> --auto --right-of <primary>`.

EEDefaults

Binary: `/usr/sbin/eeos-default-apps` **Desktop entry:** `eeos-default-apps.desktop` **Menu location:** Setup > EEDefaults

Purpose

Sets which program handles each category of file and action – browser, text editor, image viewer, PDF reader, media player, email client, file manager, archive handler, and 30+ other categories.

Architecture

A bash/gtkdialog application that scans the system for installed programs and matches them to functional categories. The core of the system is a set of small scripts in `/usr/local/bin/` named `default<category>` (e.g., `defaultbrowser`, `defaulttexteditor`, `defaultimageviewer`).

Each `default<category>` script contains a single line like:

```
exec firefox "$@"
```

EEDefaults provides a GUI to change which program each category points to.

Categories

The full list of configurable default application categories:

Category	Description	Typical default
archiver	Archive/compression tool	xarchiver
audioeditor	Audio waveform editor	mhwaveedit
audiomixer	Volume/mixer control	alsamixer (via terminal)
audioplayer	Music player	audacious
barehtmlviewer	Lightweight HTML viewer	dillo

Category	Description	Typical default
browser	Web browser	firefox or palemoon
calendar	Calendar/planner	osmo
cdplayer	CD audio player	audacious cdda://
cdrecorder	CD/DVD burner	pburn
chat	Chat/IRC client	weechat
connect	Network connection manager	connectwizard
contact	Address book	osmo
draw	Vector drawing	inkscape
email	Email client	claws-mail
filemanager	File browser	EEFiles
htmleditor	HTML/web editor	geany
htmlviewer	HTML document viewer	default browser
imageeditor	Bitmap image editor	mtpaint
imageviewer	Image viewer	viewnior
mediaplayer	Video player	vlc or mpv
paint	Paint program	mtpaint
pdfviewer	PDF reader	evince or qpdfview
processmanager	Task/process manager	lxtask or htop
screenshot	Screen capture	psnip or grim+slurp
run	Run dialog	prun
search	File search	pfind or fsearch
spreadsheet	Spreadsheet editor	gnumeric
terminal	Terminal emulator	lxterminal or urxvt
texteditor	Plain text editor	geany or l3afpad
textviewer	Text file viewer	leafpad
torrent	BitTorrent client	transmission-gtk
wordprocessor	Word processor	abiword

How It Works

1. Scans all `.desktop` files in `/usr/share/applications/` for Name, GenericName, Comment, MimeType, and Icon fields
2. For each category, builds a list of matching installed programs from:
   - A hardcoded priority list (e.g., for browser: opera, palemoon, firefox, chromium, chrome, icecat, midori, seamonkey…)
   - Dynamic search of desktop entries matching category-related keywords
3. Presents a GUI grid with dropdowns for each category
4. On save, rewrites the `/usr/local/bin/default<category>` scripts

Troubleshooting

Double-click opens the wrong program: The file association is determined by MIME type. EEDefaults sets the default program per category, but individual MIME type overrides may exist in `~/.local/share/applications/mimeapps.list`. Edit that file to fix per-filetype associations.

A program doesn't appear in the dropdown: It either isn't installed or doesn't have a `.desktop` file in `/usr/share/applications/`. Create one or install the program via EEPackages.

Part 3: Software Management

EESoftware

Desktop entry: `eeos-software.desktop` (workstation profile only) **Depends on:** gnome-software, flatpak, apt

Purpose

Graphical app store for browsing, installing, updating, and removing applications. Available in the workstation profile. Uses gnome-software as the frontend with two package sources:

1. **Flathub (Flatpak):** Sandboxed applications with their own libraries. Independent update cycle. Installed to `/var/lib/flatpak/`.
2. **Debian trixie repository (apt):** System packages that integrate directly into the OS. Installed to the root filesystem.

Flatpak vs System Packages

Flatpak apps: - Run in a sandbox with controlled access to the filesystem, network, and hardware - Include their own library dependencies (larger download, but no dependency conflicts) - Update independently of the base system - Don't affect system stability when installed or removed - Stored in `/var/lib/flatpak/app/`

System packages (apt/deb): - Integrate directly into the OS filesystem - Share libraries with the system (smaller footprint) - Can affect system stability if core libraries are upgraded - Managed via `apt-get` or `dpkg`

Configuration

Flatpak remote (Flathub) is configured at: - `/etc/flatpak/remotes.d/flathub.flatpakrepo`

Apt sources are in: - `/etc/apt/sources.list` - `/etc/apt/sources.list.d/`

Troubleshooting

EESoftware can't load: Check internet: `ping -c1 8.8.8.8`. If no network, EESoftware can't fetch package listings.

Flatpak app won't launch: Try from terminal: `flatpak run <app-id>` (e.g., `flatpak run org.gimp.GIMP`). The error output will show what's wrong – usually a missing permission or a runtime version mismatch.

Package list outdated: Run `sudo apt-get update` from a terminal to refresh the Debian package cache. Flatpak updates automatically when EESoftware is opened.

Flatpak uses too much disk space: `flatpak uninstall --unused` removes unused runtimes (shared library bundles that are no longer needed by any installed app).

EEPackages

EEOS Package Manager, version 2.5. The full-featured package manager for installing, removing, searching, and managing software packages from multiple repositories. Handles six package formats, resolves dependencies up to 13 levels deep, and supports both a graphical interface and a command-line wrapper.

This is the power-user tool. EESoftware is the friendly app store; EEPackages is what you reach for when you need to pick a specific package version, inspect a dependency tree, install something from a particular repo, or work from the command line.

Opening EEPackages

From the desktop menu: System > EEPackages.

From the terminal:

```
eeos-pkg --gui
```

The GUI launches `pkg_chooser.sh`, a gtkdialog-based interface. The window has three main zones: repository and category selectors across the top and left, the package list in the center, and your installed packages in the bottom-right.

The GUI Layout

Repository selector (top row): Radio buttons for each configured repository. The EEOS native repos appear first:

- eeos-2, eeos-3, eeos-4, eeos-5 (architecture-specific repos, versioned)
- eeos-noarch (architecture-independent packages)
- eeos-common (shared across EEOS versions)

Below those, compatible upstream repos are available:

- debian-trixie-main
- debian-trixie-universe (if configured)

Click a repo radio button to switch the package list to that repository's contents. Only one repo is active at a time in the list view, but searches can span all repos.

Category filter (left side): Radio buttons that narrow the package list by software category – All, Desktop, System, Internet, Multimedia, Graphics, Office, Development, and so on. The category mapping comes from `categories.dat` in the EEPackages installation directory.

Sub-package checkboxes: Four checkboxes labeled EXE, DEV, DOC, and NLS control which package variants appear in the list:

- **EXE** – the main executable package. This is the one you almost always want.
- **DEV** – development headers and libraries. Only needed if you're compiling software that links against this package.
- **DOC** – extended documentation. Man pages, HTML docs, examples.
- **NLS** – translations / National Language Support. Only useful if you're running EEOS in a language other than English.

By default, EXE is checked and the others are off. If you're building from source and hitting missing-header errors, check DEV. If you switched EEOS to French or another language, check NLS to pull in translated strings and menu entries.

Package list (center): Shows every package in the selected repo that matches the current category and sub-package filters. Each entry shows the package name, version, size, and a short description. Click any package to open its info popup.

Installed packages (bottom-right): Lists everything you've installed through EEPackages. This only tracks packages you added – the base system's built-in packages are listed separately in the database but don't clutter this view.

Searching for Packages

The search box is at the top of the window. Type a term and hit Enter or click Search.

Search syntax:

- **Simple name search:** Type part of the package name. The search matches from the left by default, so `fire` finds `firefox`, `firewall-config`, etc.
- **Glob wildcards:** Use `*` for wildcard matching anywhere in the name. `*codec*` finds anything with "codec" in the name. `lib*jpeg*` finds all JPEG-related libraries.
- **Multi-word search:** Type multiple words separated by spaces. The search checks both the package name and the description, so `image editor` finds packages where "image" and "editor" both appear somewhere in the name or description fields.
- **Case-insensitive:** All searches ignore case. `Firefox`, `firefox`, and `FIREFOX` all return the same results.

Search scope: By default, the search runs against the currently selected repository. There's an option to search across all configured repos at once – use this when you're not sure which repo has the package you need.

Examples:

You type	What it finds
`vlc`	VLC media player and related packages in the current repo
`*codec*`	All packages with "codec" anywhere in the name
`image viewer`	Packages where both "image" and "viewer" appear in the name or description
`lib*png*`	All PNG-related libraries
`python3`	Everything starting with "python3"

Installing a Package

1. Find the package you want (browse or search).
2. Click the package name in the list.
3. An info popup appears showing:
   - Package name, version, and size
   - Full description
   - The dependency tree – every package this one needs, displayed as a hierarchy
   - Which dependencies are already installed (marked) and which are missing
4. Click Install (or the equivalent confirmation button).
5. EEPackages downloads the package and all missing dependencies, then installs everything in the correct order.

What happens during installation:

The install handler (`installpkg.sh`) does the following for each package:

1. Checks available memory. If `/tmp` is more than 90% full, the install aborts with a warning rather than risking a partial extraction.
2. Downloads the package to the configured download directory (default `/root`).
3. Extracts the package to a temporary work directory.
4. Handles usrmerge if needed – modern Debian-style packages expect `/bin` to be a symlink to `/usr/bin` . If EEOS is running with usrmerge, the installer routes files accordingly.
5. Cleans up whiteout files from the overlay filesystem so removed files from previous operations don't interfere.
6. Runs the package's post-install script (`pinstall.sh`) if one exists. This is where the package does its own setup – creating config files, registering services, etc.

7. Updates `.desktop` files: fixes the Categories field so the app appears in the right menu section, cleans up the Exec line, and sets the icon path.
8. If the package installed GIO modules, runs `gio-query-modules` to rebuild the module cache.
9. If the package installed GLib schemas, runs `glib-compile-schemas` to compile them.
10. If a language pack is installed for the current locale, applies translations to any new `.desktop` files so menu entries appear in the user's language.
11. Runs `z_update_system_cache.sh` to rebuild system caches (icon cache, mime database, font cache, etc.) so the new software integrates cleanly.

After installation, the package is recorded in the user-installed-packages database so EEPackages can track it, show it in the installed list, and cleanly remove it later.

Install Modes

EEPackages offers three installation modes, selectable from the install dialog:

1. **Auto install (default):** Downloads and installs the package and all dependencies silently. Progress shows in the status area but you don't need to confirm each step. This is the right choice 95% of the time.
2. **Download packages (no install):** Downloads the package files to your download directory but doesn't install them. Use this when you want to grab packages on a machine with internet and transfer them to an offline machine, or when you want to inspect the packages before installing.
3. **Classic step-by-step:** The traditional mode. Every step gets its own confirmation dialog – download, extract, install, post-processing. Full visibility into exactly what's happening. Use this when troubleshooting a package that's giving you problems, or when you want to see exactly what files a package drops where.

Removing a Package

Click the Installed tab or the installed packages panel in the bottom-right. Select the package you want to remove. You get a removal preview showing:

- The files that will be removed
- Any other packages that depend on this one (so you don't accidentally break something)
- The removal mode options

Removal modes (from `removemodes.sh`):

EEPackages shows a preview before doing anything destructive. You see exactly what's going to be deleted and what might break. Confirm to proceed, or cancel out.

Dependency Resolution

This is where EEPackages earns its keep. The dependency resolver (`dependencies.sh`) does recursive resolution up to 13 levels deep.

How it works:

1. The resolver reads the target package's dependency list from the database. Dependencies are marked with a `+` prefix in the dependency field.
2. For each dependency, it checks whether it's already installed (present in the installed-packages database).

3. For missing dependencies, it searches the current repo first, then falls back to other configured repos.
4. Each missing dependency's own dependencies are then resolved the same way – recursively, up to 13 levels.
5. Version constraints are respected. The database supports version operators (`&ge` for greater-or-equal, `&lt` for less-than, etc.), so if a package needs `libfoo >= 2.0`, the resolver won't accept `libfoo 1.9`.
6. The full dependency tree is displayed visually in the install preview, so you can see exactly what's being pulled in and why.
7. A progress indicator shows how many missing dependencies have been found as the resolver works through the tree.

Cross-repository resolution: If a dependency can't be found in the current repo, the resolver checks the other configured repos. This means a package from eeos-3 can pull a dependency from debian-trixie-main transparently.

If the resolver can't find a required dependency in any repo, it tells you what's missing so you can decide whether to proceed without it or find the package another way.

Package Formats

EEPackages handles six package formats:

Format	Extension	Notes
EEOS native	`.pet`	The native format. May be split into `_DEV`, `_DOC`, `_NLS` sub-packages.
Debian	`.deb`	Full support. This is the most commonly used format for upstream packages.
Slackware	`.tgz`, `.txz`	Converted and installed. Good for packages only available in the Slackware ecosystem.
RPM	`.rpm`	Converted to a compatible format and installed.
T2	`.tar.bz2`	T2 Linux packages, extracted and installed.
SquashFS	`.sfs`	Not extracted – overlaid as a filesystem layer. Used for large package bundles or system modules.

You can also install `.pet` and `.deb` files directly from the file manager by clicking on them. The system's file handler routes them to EEPackages' install handler automatically.

The Package Database

Package metadata lives in `/root/.packages/`. The database files use a pipe-delimited format:

```
pkgname|version|build|category|size|path|filename|dependencies|description|repo
```

Key database files:

- `eeos-installed-packages` – packages that came with the base system. These are the built-in packages from the SFS layers. Don't manually edit this.
- `user-installed-packages` – packages you installed through EEPackages. This is how the tool tracks what to show in the "Installed" view and what can be cleanly removed.
- `Packages-*` – repo package lists. One file per configured repository. These are downloaded and updated when you refresh the database.
- `DISTRO_PKGS_SPECS` – the master package specification for the EEOS build. Lists what's expected to be in the base system.
- `DISTRO_EE_REPOS` – repository URLs and configuration.
- `PKGS_MANAGEMENT` – internal management metadata.

Database conversion: The `debdb2eeosdb` tool converts Debian repository package lists into the EEOS pipe-delimited format. This is what makes Debian repos available inside EEPackages without needing a separate package manager.

Database updates: When you switch repos or the tool detects stale data, it can re-download and rebuild the package lists. The "No user input during database updating" config option controls whether this happens silently or with confirmation prompts.

Configuration

Open the configuration dialog from the EEPackages menu, or it shows on startup if you've enabled that option.

Available settings:

Option	What it does
Skip package size check when >4GB storage available	Skips the free-space verification before installing. Only safe when you have plenty of room.
Custom download path	Changes where downloaded package files are saved. Default is `/root`.
Install to savefile instead of directly	Routes installations into the session save file rather than the live overlay. Only available in save mode 13. Use this if you want installed packages to survive a reboot without doing a session save afterward.
Show/hide terminal during activity	Controls whether a terminal window pops up showing download progress, extraction output, etc. Hide it for a cleaner experience; show it when debugging.
No user input during database updating	

Option	What it does
	When the package database is refreshed, skip all confirmation dialogs and just do it. Useful for automation or if the prompts annoy you.
Redownload already-downloaded packages	Forces a fresh download even if the package file already exists locally. Use this if you suspect a corrupted download.
Don't delete packages after installation	Keeps the downloaded `.deb` / `.pet` files around after installation instead of cleaning them up. Useful if you want to save them for offline installation on another machine.
Show config options on startup	Pops the configuration dialog every time EEPackages launches, so you can adjust settings before doing anything.

Command-Line Interface

The `eeos-pkg` CLI wrapper provides quick access without the GUI:

```
eeos-pkg install <package>    # install a package by name
eeos-pkg remove <package>     # remove an installed package
eeos-pkg search <term>        # search for packages matching a term
eeos-pkg list                 # list all installed packages
eeos-pkg --gui                # launch the graphical interface
```

Examples:

```
eeos-pkg search image editor
eeos-pkg install gimp
eeos-pkg remove gimp
eeos-pkg list | grep python
```

The CLI is a thin wrapper. For complex operations (dependency trees, cross-repo installs, sub-package filtering), use the GUI.

Package Integrity Verification

EEPackages includes `verifypkg.sh` for checking downloaded packages before installation. This verifies that the package file isn't corrupted or truncated. The verification runs automatically during the install process – you don't need to invoke it separately unless you're doing manual troubleshooting.

The `testurls.sh` tool tests download URLs to verify that mirrors are reachable and serving the expected files. This is used internally when mirror selection happens and when troubleshooting download failures.

How EEOS Updates Work

EEOS doesn't update by replacing packages one at a time like a traditional Linux distribution. Instead, the entire system image is updated by replacing the SFS files (the read-only system layers). Individual packages you've installed through EEPackages are preserved in your session save and overlaid on top of the new base image.

This means:

- Base system updates are atomic – the new SFS either works or you boot the old one.
- Your user-installed packages carry over automatically.
- You don't need to run `apt upgrade` or worry about partial update states.
- If an update breaks something, remove the new SFS and boot the previous version.

EEPackages is for adding software on top of the base system, not for updating the base system itself.

Installing Packages Without the GUI

You don't need to open EEPackages to install a `.pet` or `.deb` file you've already downloaded. Just click the file in the file manager. The system's MIME handler routes it to the EEPackages install handler, which shows a confirmation dialog and installs it.

This also works from the command line:

```
eeos-pkg install /path/to/package.deb
```

Help System

EEPackages has a built-in help system accessible from the Help menu inside the GUI. Documentation is available in English (`help.htm`) and French (`help-fr.htm`). The help covers the same topics described here: search syntax, installation workflow, repo system, and sub-package types.

Architecture

For reference, EEPackages consists of 30+ shell scripts in `/usr/local/eeos-pkg/` :

Script	Purpose
`pkg_chooser.sh`	Main GUI (gtkdialog)
`installpkg.sh`	Package installation handler
`dependencies.sh`	Dependency resolver (13-level recursive)
`downloadpkgs.sh`	Package downloader with mirror selection
`configure.sh`	Configuration dialog
`filterpkgs.sh`	Package list filtering
`postfilterpkgs.sh`	Post-filter processing

Script	Purpose
`findnames.sh`	Package search engine
`finduserinstalledpkgs.sh`	Tracks user-installed packages
`findmissingpkgs.sh`	Finds missing dependencies
`removepreview.sh`	Uninstall preview
`removemodes.sh`	Uninstall mode selection
`installpreview.sh`	Install preview
`installmodes.sh`	Install mode selection
`check_deps.sh`	Dependency checking
`check_deps_gui.sh`	Dependency checking (GUI version)
`installed_size_preview.sh`	Shows disk usage before install
`verifypkg.sh`	Package integrity verification
`testurls.sh`	Download URL testing
`fetchinfo.sh`	Package metadata fetcher
`categories.dat`	Category mapping data
`debdb2eeosdb`	Debian-to-EEOS database converter
`hacks-postinstall.sh`	Post-install fixups
`hacks-postinstall2.sh`	Additional post-install fixups
`z_update_system_cache.sh`	System cache rebuild
`0setup`	Repository setup (from build system)
`help.htm`	English help
`help-fr.htm`	French help

Troubleshooting

Package won't install – dependency errors: Open EEPackages, find the package, and look at the dependency tree in the info popup. The tree shows which dependencies are missing and which repos were searched. If a dependency can't be found in any repo, you may need to add a repo that carries it, or find and install the dependency manually.

Installation fails with "not enough space": EEPackages checks available space before extracting. If `/tmp` is over 90% full, it aborts. Free up space in the overlay or session save, or boot with more RAM allocated to the tmpfs.

Downloaded package is corrupted: Enable "Redownload already-downloaded packages" in the configuration, then try the install again. If it still fails, test the mirror URL with `testurls.sh` or try a different repository that carries the same package.

Package installed but doesn't appear in the menu: The post-install processing should update the desktop database and menu cache. If it didn't, run `update-menus` from a terminal, or log out and back in. Check that the package's `.desktop` file has a valid Categories field.

Installed package broke something: Open EEPackages, go to the installed packages view, select the problematic package, and remove it. The removal preview shows you what will be deleted. If the system is too broken to open the GUI, use the CLI: `eeos-pkg remove <package>`.

Database is stale or empty: Switch to a repo in the GUI and let the database refresh. If it fails, check network connectivity. The database files in `/root/.packages/` can be deleted and re-downloaded – they're just cached repo metadata, not records of your installed packages.

Can't find a package you know exists: Check that you're searching the right repo. Try the "search all repos" option. Check that EXE is checked in the sub-package filter – if only DEV is checked, you won't see the main packages. Try a glob search with wildcards: `*packagename*`.

EEDrivers

Binary: `/usr/sbin/eeos-driver-manager` (workstation profile) **Desktop entry:** `eeos-driver-manager.desktop` **Menu location:** System > EEDrivers

Purpose

GPU driver auto-detection and installation. Identifies the graphics card and offers the appropriate driver package.

GPU Detection

EEDrivers uses `lspci` to identify the GPU vendor and model:

- **NVIDIA:** Offers the proprietary `nvidia-driver` package. Installation requires a reboot. If the screen goes black after install, the "NVIDIA Safe" GRUB entry (which disables `nouveau`) provides a safe fallback.
- **AMD:** Offers the `firmware-amd-graphics` package for the open-source `amdgpu` driver. Usually works without reboot.
- **Intel:** Offers `i965-va-driver-shaders` or `intel-media-va-driver-non-free` for hardware video acceleration (VA-API). The built-in `i915` kernel driver handles rendering.

Troubleshooting

Screen black after NVIDIA driver install: Boot with "NVIDIA Safe" from the GRUB menu. From CLI: `sudo apt remove nvidia-driver` to uninstall.

AMD GPU artifacts or glitches: Check that the firmware is loaded: `dmesg | grep amdgpu`. Missing firmware files cause rendering issues. The `eeos_firmware.sfs` should contain them.

Intel video tearing: Enable TearFree: create `/etc/X11/xorg.conf.d/20-intel.conf`:

```
Section "Device"
    Identifier "Intel"
    Driver "intel"
    Option "TearFree" "true"
EndSection
```

Part 4: Session and Storage

EESession

Binary: `/usr/local/bin/eeos-save-manager` **Related:** `/usr/sbin/eeos-make-save`, `/usr/local/bin/eeos-session-merge`, `/usr/sbin/eeos-savefolder-to-savefile`, `/usr/sbin/eeos-save-to-sfs` **Desktop entry:** `eeos-save-manager.desktop` **Menu location:** System > EESession

Purpose

Manages persistent storage for EEOS. Since EEOS runs from a read-only squashfs image with a RAM overlay, changes are lost on shutdown unless captured in a save file or save folder on the boot USB.

How Persistence Works

EEOS uses a layered filesystem: 1. **Base layer:** Read-only SFS modules (eeos_system.sfs, eeos_base.sfs, etc.) 2. **Overlay:** RAM-based tmpfs where all runtime changes go 3. **Save layer:** Optional persistent storage on the USB that captures the overlay between sessions

On shutdown, the overlay contents (new files, modified configs, installed packages) are written to the save layer. On next boot, the save layer is loaded back into the overlay.

Save Types

SAVE FILE (EEOSSAVE.4FS)

A single file on the USB stick containing an ext4 filesystem image. - **Pros:** Works on FAT32, NTFS, exFAT, and ext4 USB filesystems. Portable. - **Cons:** Fixed size (must be resized if it fills up). Single file can be slow on FAT32. - **Default size:** Created at 512MB, expandable.

SAVE FOLDER (EEOSSAVE/)

A directory on the USB stick where files are stored directly. - **Pros:** Grows dynamically (no fixed size limit). Faster file access. Easier to browse and recover individual files. - **Cons:** Only works on Linux filesystems (ext4, f2fs, ext3). The USB must be formatted with a Linux FS.

Operations

CREATE

Creates a new save file or save folder on the boot USB. Asks for: - Save type (file or folder) - Size (for save file only) - Location on the USB

RESIZE

Expands a save file when it's running low on space. Shows current usage and asks for new size. Cannot shrink – only grow.

MERGE

Combines the current overlay with the save to reclaim space. Over time, the overlay accumulates deleted file markers and modified file copies that waste space. Merging collapses these into a clean state.

CONVERT

Switches between save file and save folder formats. Copies the contents from one format to the other.

Related Tools

- `eeos-session-merge`: CLI tool for merging the overlay into the save. Can be run from terminal. Has an `.overlay` variant for overlay-specific merge operations.
- `eeos-make-save`: Low-level tool for creating save files/folders.
- `eeos-savefolder-to-savefile`: Converts a save folder to a save file.
- `eeos-save-to-sfs`: Captures the current save state as an SFS module (useful for creating a baseline that can be loaded as a read-only layer).

Troubleshooting

Changes not persisting: 1. Check boot mode: if you booted with "RAM only (ephemeral)", persistence is intentionally disabled 2. Check that a save file/folder exists: look for `eeossave.4fs` or `eeossave/` on the USB 3. Check free space on the save: `df -h` in EEDiagnostics > Storage 4. Try merging the session to reclaim space

Save file is full: Open EESession > Resize. If resize fails due to lack of space on the USB, you need a larger USB or need to delete files from the system.

Merge fails: Must be done in Standard mode, not RAM-only mode. The merge needs write access to the save layer.

Convert fails: Ensure the target filesystem supports the desired save type. FAT32 can't hold a save folder.

EEBackup

Session save backup and restore tool, version 1.3.

Binary: `/usr/sbin/eeos-session-backup` **Desktop entry:** `eeos-session-backup.desktop` **Menu location:** System > EEBackup **Lock file:** `/tmp/eeos-session-backup-lock` (prevents multiple instances)

What It Does

Creates timestamped backup copies of your live session save file or folder. Also restores from those backups. This is your safety net before system changes – if something goes wrong, you can roll back to a known good state.

Only one instance can run at a time. If you try to open it while it's already running, you'll get an error.

When to Use It

- Before installing new packages or updating existing ones
- Before running EERemaster
- Before making significant configuration changes (display, network, boot)
- Before testing anything you're not 100% sure about
- Periodically, as general insurance on a machine that's been running well

---

Boot Mode Requirement

EEBackup only works in save modes 12 and 13 – frugal installs with persistence. It reads `/etc/rc.d/EE0SSTATE` to detect the current boot mode. If you're running in RAM-only mode, copy-to-RAM mode, or a full install, EEBackup won't have a session save to back up and will tell you so.

Launch Dialog

When you open EEBackup, a dialog asks you to choose:

- **EEOS Session Backup** – create a new backup
- **EEOS Session Restore** – restore from an existing backup
- **Cancel** – close without doing anything

Backup Operation

What happens when you click "EEOS Session Backup":

1. A warning dialog appears: "Shut down any running applications and servers before backing up." This matters because active programs may have unsaved state or lock files that could create an inconsistent backup.
2. The tool locates your active session save by reading `EE0SSTATE`. The save file or folder lives in the install directory (typically `/mnt/home` plus any subdirectory configured during installation).
3. It creates a copy with a timestamp appended to the filename:

   ```
   yoursavename.BKP–2026.04.06–14.30
   ```

   The format is `.BKP–YYYY.MM.DD–HH.MM`, so backups sort chronologically and you can tell at a glance when each was made.
4. A progress bar shows the copy operation with elapsed time.
5. When done, a confirmation dialog reports the backup location.

Scope limitation: EEBackup backs up your system configuration *as it exists in the session save*. It does not back up data stored outside the session save – files on other partitions, mounted network drives, external USB storage, or anything under `/mnt` that isn't part of the save layer. The built-in help text makes this explicit.

Restore Operation

What happens when you click "EEOS Session Restore":

1. An explanation screen appears describing what the restore can handle: compressed or uncompressed save files and folders. Backups must be in the same directory as the active save file – the restore tool won't search other locations.
2. The tool reads `EEOSSTATE` to find the install directory and lists all `.BKP` files found there. These appear in a list dialog where you select which backup to restore.
3. An optional "Add To Name" dialog lets you add a suffix to the restored item's name. Useful if you're restoring multiple backups to compare them.
4. The tool detects the backup type and handles it accordingly:
    - **Uncompressed saves** (`.2fs` , `.3fs` , `.4fs` extensions): The tool detects the filesystem type and performs a direct copy with a progress bar showing elapsed time.
    - **Compressed saves** (`.tar` archives): The tool extracts to a temporary "extracted" directory, detects whether the contents are a single file or folder, then moves them into place.
5. The restored item is renamed to the format:

    ```
    yoursavename-2026.04.06-RESTORED.4fs
    ```

    Or with a custom suffix:

    ```
    yoursavename-2026.04.06-RESTORED-mysuffix.4fs
    ```

6. A final dialog tells you to reboot to use the restored session.

Manual restore alternative: You can also restore manually by finding the backup in a file manager and removing the `.BKP-YYYY.MM.DD-HH.MM` extension from the filename. The boot parameter `pfix=psavebkp` forces the init system to recognize backup-created saves that it would otherwise skip.

Help System

Both the backup and restore screens have built-in help accessible from the GUI. The help text explains:

- What gets backed up (only the session save contents)
- What doesn't get backed up (data outside the save)
- How to restore manually
- The `pfix=psavebkp` boot parameter for edge cases

File Paths

File/Path	Purpose
`/usr/sbin/eeos-session-backup`	Main script (697 lines, bash/yad/gtkdialog)
`/etc/rc.d/EEOSSTATE`	Boot mode and save file location
`/tmp/eeos-session-backup-lock`	Instance lock file
`/mnt/home$PSUBDIR/`	Default location for save files and backups

Troubleshooting

"Not available in this boot mode": - You're not running a frugal install with persistence. EEBackup requires save modes 12 or 13. Check your boot mode with `cat /etc/rc.d/EEOSSTATE`. - If you're running in copy-to-RAM mode, changes aren't persisted anyway – there's nothing to back up.

Backup takes a very long time: - Large session saves (1GB+) take proportionally longer. The progress bar shows elapsed time so you can gauge progress. - If the save is on a slow USB stick, the bottleneck is I/O. Consider running EESession to merge and shrink the save before backing up.

No backups appear in the restore list: - Backups must be in the same directory as the active save file. If you moved backups to another location, move them back. - Check that the backup files have the `.BKP-` extension pattern. Files with other naming won't be detected.

Restore completed but system hasn't changed: - You need to reboot after restoring. The restored save isn't used until the next boot. - If you rebooted and still see the old state, verify the restored file was named correctly. The init system looks for specific filename patterns.

"Already running" error on launch: - A stale lock file exists at `/tmp/eeos-session-backup-lock`. If you're sure no other instance is running, delete it: `rm /tmp/eeos-session-backup-lock`.

EESwap

Swap file management.

Source: `/usr/sbin/eeos-swap` (370 lines, bash) **Desktop entry:** `eeos-swap.desktop` **Menu location:** System > EESwap

What it does: Creates, configures, activates, and manages swap files for systems that need more virtual memory than physical RAM provides.

When to Use Swap

EEOS typically runs without swap because it operates from RAM or an overlay filesystem. You need swap when:

- The machine has less than 4GB RAM and runs heavy desktop applications
- You're compiling software (gcc can eat a lot of memory)

- You're editing large images or spreadsheets
- EEDiagnostics reports RAM usage consistently over 85%
- Applications are being killed by the OOM (Out Of Memory) killer

If none of these apply, don't bother. Swap on flash storage shortens the drive's lifespan and is significantly slower than RAM.

Operations

Create a swap file:

1. Choose the location. Must be on a Linux filesystem – ext4, f2fs, etc. FAT32 and NTFS don't support the swap file format and will fail silently or throw errors.
2. Set the size in MB. Rules of thumb:
   - Under 4GB RAM: set swap equal to RAM size
   - 4GB or more: set swap to half of RAM
   - Compiling large projects: 2x RAM if you have the disk space
3. The tool creates the file, sets permissions to 600 (root-only), runs `mkswap` to format it, and reports success or failure.

Activate swap:

Runs `swapon` on the swap file. Takes effect immediately – the kernel starts using it for overflow memory the moment it's active. No reboot needed.

Deactivate swap:

Runs `swapoff`. Everything currently in swap gets moved back to RAM. If RAM is too full to absorb the swap contents, the kernel will start killing processes to make room. Don't deactivate swap on a system that's already under memory pressure unless you're prepared for that.

Remove swap file:

Deactivates the swap (if active) and deletes the file. Only works when the swap isn't pinned by processes that would crash without it.

Resize:

There's no in-place resize. To change swap size: deactivate, delete, create a new file at the desired size.

Swap File vs Swap Partition

EESwap creates swap **files**, not swap partitions. This is deliberate – files are more flexible:

- Can be created on any existing Linux partition with free space
- Can be "resized" by creating a new file
- Don't require repartitioning or any changes to the disk layout
- Work with the EEOS overlay and persistence system
- Easy to remove completely when no longer needed

Persistence

If you create the swap file on the boot USB's Linux partition (the persistent storage area), it survives reboots. If you create it in the tmpfs overlay (the default writable layer when persistence isn't configured), it disappears on shutdown.

For center deployments running HHFE, you generally don't need swap unless the machine has very little RAM (under 2GB). The EEAI Boot Profile Selector recommends CLI-only mode for machines under 2GB, which avoids the memory pressure that would require swap in the first place.

Troubleshooting

"swapon: invalid argument" error:

The file wasn't formatted with `mkswap`, or it's on a FAT32/NTFS filesystem. FAT32 doesn't support the sparse file structure that swap requires. Create the swap file on an ext4 or f2fs partition instead.

System still runs out of memory with swap enabled:

Swap is orders of magnitude slower than RAM. If the system is constantly swapping (check with `free -h` or `vmstat 1` – look for high `si`/`so` values), adding more swap won't help. The machine needs more physical RAM, or you need to reduce memory usage by closing applications or switching to a lighter boot profile.

Swap file disappeared after reboot:

It was created in the tmpfs overlay or on a partition that isn't mounted at boot. Create it on the boot USB's persistent partition, which is mounted early in the boot process.

Command-line equivalents:

```
# Check current swap status
free -h
cat /proc/swaps

# Manual swap creation (what EESwap does under the hood)
dd if=/dev/zero of=/mnt/sda1/swapfile bs=1M count=2048
chmod 600 /mnt/sda1/swapfile
mkswap /mnt/sda1/swapfile
swapon /mnt/sda1/swapfile

# Manual deactivation
swapoff /mnt/sda1/swapfile
```

Part 5: Installation and Imaging

EEInstaller

Full drive installer for EEOS.

Binary: `/usr/sbin/eeos-installer-classic` (1,155 lines, bash) **Wrapper:** `/usr/local/bin/eeos-installer` **Desktop entry:** `eeos-installer.desktop` **Menu location:** System > EEInstaller

---

What It Does

Installs EEOS to internal drives or USB sticks. Copies the SFS layers, kernel, initrd, and boot loader configuration to the target device and makes it bootable. This is the tool you use to put EEOS on a machine permanently or to create a bootable USB with full persistence.

Installation Types

Frugal Install (recommended)

Copies the EEOS system files – SFS modules, kernel, initrd, boot configuration – to a directory on an existing partition. The OS runs from those files with a writable overlay for changes. This is the standard EEOS install method and the one you should use unless you have a specific reason not to.

Why frugal is better for most cases: - Multiple EEOS versions can coexist on the same partition. Just install them into different directories. - The system files are a handful of files in one directory – back them up by copying the directory, delete the installation by deleting the directory. - Updating EEOS means replacing those files, not reformatting the drive. - Session save files sit alongside the system files, making everything self-contained. - The layered SFS architecture stays intact, so you keep the ability to load/unload modules.

Full Install

Extracts the SFS contents onto a partition as a traditional Linux filesystem. The entire root tree is laid out directly on disk. This loses the layered SFS architecture, the module system, the easy update path, and the ability to roll back by swapping SFS files. Not recommended for most deployments. The only real use case is when you need direct filesystem access to every system file without the overlay, which is rare.

The Installation Process (Frugal)

1. **Select target drive.** The installer lists all detected drives – internal HDDs/SSDs and USB sticks – with model name, size, and bus type. Pick the one you want to install to.
2. **Choose or create a partition.** If the target drive already has partitions, select one. If it doesn't, or if you want to repartition, the installer can launch a partition editor. For a clean install, one partition using the full drive is the simplest setup.
3. **Select installation type.** Frugal (recommended) or Full.
4. **The installer copies the system files:**
   - All SFS modules (`eeos_system.sfs` , `eeos_base.sfs` , `eeos_modules.sfs` , and any others)
   - The kernel (`vmlinuz`)
   - The initial ramdisk (`initrd.gz`)
   - Boot loader files (GRUB2 or syslinux, depending on target and UEFI/BIOS)
   - Boot configuration with all EEOS boot entries

5. **Boot loader configuration.** The installer writes entries for every EEOS boot mode:
    - Standard boot
    - Copy to RAM
    - CLI only
    - Recovery Shell
    - Safe Video (nomodeset)
    - NVIDIA Safe
    - Maximum Compatibility
    - EEAI Assistant (smart profile selection)
6. **Session save creation.** An initial save file or save folder is created for persistence. On ext4 partitions, a save folder is the default. On FAT32, a save file is used (FAT32 doesn't support the extended attributes needed for folders).

Partition Requirements

Filesystem	Save Type	Max Save Size	Notes
ext4	File or folder	Unlimited	Recommended. Save folders have no size limit and better performance.
f2fs	File or folder	Unlimited	Good for flash storage (USB sticks, SD cards).
ext3	File or folder	Unlimited	Legacy. Works fine, no reason to prefer it over ext4.
FAT32	File only	4GB	Folders not supported. The 4GB file size limit constrains the save.
NTFS	File or folder	Unlimited	Works but not ideal. NTFS-3G driver adds overhead.

Minimum free space: ~1GB for the system files, plus whatever you want for the session save. A comfortable minimum is 4GB total.

UEFI vs BIOS

The installer detects whether the machine booted in UEFI or BIOS mode and configures the boot loader accordingly:

- **BIOS boot:** Uses syslinux for USB targets, GRUB2 for internal drives.

- **UEFI boot:** Uses GRUB2 with an EFI System Partition. If the target drive doesn't have one, the installer creates it.

On USB sticks, the installer tries to make the stick bootable in both modes when possible, so the same USB works on UEFI and BIOS machines.

USB vs Internal Drive

USB sticks: - The installer uses syslinux (BIOS) or GRUB (UEFI) depending on the detected boot mode. - FAT32 is common on USB sticks but limits the save to 4GB. If you need more, format the stick as ext4 first. - For center deployments where USBs are the primary boot media, use the EEStick Installers instead – they're purpose-built for that workflow.

Internal drives: - Always uses GRUB2 regardless of boot mode. - Installing to an internal drive modifies that drive's partition table and boot loader. The installer shows the drive model and size before confirming, so verify you're targeting the right drive. - On machines that already have another OS, the installer adds EEOS boot entries to GRUB alongside the existing OS (dual-boot). It does not overwrite the other OS's partition unless you explicitly select that partition as the target.

File Paths

File/Path	Purpose
`/usr/sbin/eeos-installer-classic`	Main installer script (1,155 lines, bash)
`/usr/local/bin/eeos-installer`	Wrapper script (19 lines)
`eeos-installer.desktop`	Desktop menu entry
`/etc/DISTRO_SPECS`	Distribution version info (read by installer)
`/etc/rc.d/EEOSSTATE`	Boot mode detection

Troubleshooting

Installer doesn't detect the target drive: - USB sticks: unplug and replug, then relaunch. Some USB controllers need a moment after hotplug. - Internal drives: check that the drive appears in `lsblk`. If it doesn't, the drive may not be connected properly or the controller driver isn't loaded (check EEKView).

Boot loader installation fails: - On UEFI systems, the EFI System Partition may be too small or mounted read-only. The installer needs write access to the ESP. - On BIOS systems with GPT disks, a BIOS boot partition (1MB, unformatted, `bios_grub` flag) is required. The partition editor can create one.

System won't boot after installation: - Enter the BIOS/UEFI setup and verify the boot order includes the target drive. - On UEFI machines, Secure Boot may need to be disabled. EEOS doesn't ship with signed boot loaders. - Try booting from the original EEOS media and using the Recovery Shell to inspect the target drive.

Save file not created or too small: - On FAT32, the max save file is 4GB. If you need more, reformat as ext4. - If the save wasn't created at all, the partition may have been mounted read-only during installation. Check `mount` output.

"Not enough space" error: - The system files need ~1GB. Check `df -h` on the target partition. - On USB sticks, this usually means the stick is smaller than expected or has hidden partitions consuming space.

For center deployments: - Use the EEStick Installers instead of EEInstaller. They're optimized for USB-only workflows and handle the center-specific boot profiles automatically.

EERemaster

Build a custom EEOS system image from the running system.

Binary: `/usr/sbin/eeos-remaster-classic` (380 lines, sh/Xdialog) **Alternative method:** `/usr/sbin/eeos-remaster-alternative` (378 lines) **Wrapper:** `/usr/local/bin/eeos-remaster` **Desktop entry:** `eeos-remaster-classic.desktop` **Menu location:** System > Remaster EEOS

What It Does

Takes everything on the running EEOS system – every package you've installed, every config you've changed, every file you've added – and compresses it into a new SFS image. Optionally builds a bootable ISO from that image. The result is a distributable, deployable snapshot of the exact system you're running right now.

This is how you create standardized center deployments, recovery images, and pre-configured workstation setups. Get one machine right, remaster it, and deploy that image to every other machine.

Architecture

EERemaster uses Xdialog for its GUI. It reads `/etc/rc.d/EEOSSTATE` for the current boot mode and `/etc/DISTRO_SPECS` for version and naming. The core operation is running `mksquashfs` against the live root filesystem with a carefully curated exclusion list, then optionally wrapping the result in a bootable ISO.

Compression defaults to gzip with 16384 block size. The compression progress is displayed in an rxvt terminal window so you can watch it work.

There are two methods available: the "classic" remaster and an "alternative" method. Both produce the same output – a new SFS and optionally an ISO. The wrapper at `/usr/local/bin/eeos-remaster` launches the classic method by default.

The Checklist Dialog

When you launch EERemaster, a dialog presents six options as a checklist. Some are on by default, some are off. Here's what each one does:

#	Option	Default	What It Does
1		Off	

#	Option	Default	What It Does
	Choose extra SFS files to include		Opens a secondary dialog where you can include or exclude individual loaded SFS modules from the remastered image. By default, all loaded extra SFS files are included. Turn this on only if you want to leave some modules out.
2	Recording changes to system	**On**	**Do not deselect.** This ensures your overlay changes (installed packages, config edits, added files) are captured in the new image. Turning it off would produce an image identical to the base SFS, which defeats the purpose.
3	Discard current hardware settings	Off	Strips hardware-specific configuration from `/etc` in the new image. Select this when you're building an image that will be deployed on different hardware than the machine you're remastering on. Leaves the image hardware-agnostic so it auto-detects on first boot.
4	Building new base SFS	**On**	**Do not deselect.** This is the actual squashfs build step. Without it, nothing gets built.
5	Select to also make an ISO	Off	After building the SFS, wraps it with the kernel, initrd, and boot loader into a bootable ISO. Turn this on if you want a file you can write directly to USB or burn to optical media.
6	(reserved)	–	Not used in current version.

For most remaster jobs, leave options 2 and 4 on, everything else off. Add option 5 if you want an ISO. Add option 3 if the target machines differ from the build machine. Add option 1 only when you need fine-grained control over which SFS modules are baked in.

Working Area Selection

Squashfs compression needs a working partition with enough free space to hold the temporary build files. After the checklist, EERemaster shows all available partitions with:

- Filesystem type
- Total size
- Free space available

Pick any partition with sufficient room. Read-only partitions are automatically excluded from the list. If you have enough RAM, tmpfs (the in-memory filesystem) may also appear as an option – it's fast but requires the system to have free RAM roughly equal to the final SFS size.

How much space do you need? Roughly 1.5x the expected SFS output size. A typical EEOS system image runs 400-700MB, so you'd want at least 1GB free on the working partition. If you've installed a lot of packages, the image could be larger – plan accordingly.

The Build Process (Step by Step)

1. **SFS module selection** (if option 1 was checked): Opens the `sfs_load` interface where you can toggle individual loaded SFS modules on or off. Anything toggled off won't be included in the remastered image.
2. **Firmware and kernel module preservation:** The remaster copies firmware files and kernel modules from the base layer into the build area. This ensures the new image has all the hardware support from the original base, even if you haven't loaded those modules in the current session.
3. **Working directory setup:** Creates a build directory on the partition you selected.
4. **Squashfs compression:** Runs `mksquashfs` against the root filesystem. This is the long step – expect several minutes depending on system size and CPU speed. The exclusion and inclusion logic is detailed below.
5. **Post-processing:** Copies `/root` to a temp area and merges it into the SFS. Selectively copies `/etc` (some things are runtime-generated and shouldn't persist). Selectively copies `/var`.
6. **Hardware settings removal** (if option 3 was checked): Strips hardware-specific config from `/etc` – things like X11 device configurations, network interface names, and GPU-specific settings.
7. **ISO build** (if option 5 was checked): Packages the new SFS with the kernel (`vmlinuz`), initial ramdisk (`initrd.gz`), and a boot loader into a bootable `.iso` file.

What Gets Excluded (and Why)

The squashfs build excludes these paths from the root filesystem:

Excluded Path	Reason
`/proc`	Virtual filesystem, kernel-generated at runtime. Has no on-disk content.
`/sys`	Virtual filesystem for hardware info. Kernel-generated.
`/tmp`	Temporary files. Cleared on boot.
`/mnt`	Mount points for other devices. Should be empty in the image.
`/media`	Auto-mount points. Should be empty in the image.
`/home`	User data directory. Not part of the system image.
`/initrd`	Init ramdisk mount point. Runtime only.
`/var`	Partially excluded, then selectively re-added (see below).
`/etc`	Partially excluded, then selectively re-added (see below).
`/dev/snd/*`	Sound device nodes. Created at runtime by udev.
`/dev/.udev`	Udev runtime data. Regenerated on boot.
Icon caches	`.icon-cache` files. Regenerated from the installed icon themes.

After the main exclusion pass, EERemaster adds back:

Added Back	How
`/proc`	Empty directory structure only (keep-as-directory)
`/tmp`	Empty directory structure only (keep-as-directory)
`/mnt`	Empty directory structure only (keep-as-directory)
`/root`	Copied to a temp area, cleaned, then merged into the SFS
`/etc`	Selectively – runtime-generated files are stripped, persistent config is kept
`/var`	Selectively – log files and caches are stripped, important state is kept

The result is a clean image that contains the full system but none of the runtime detritus. On first boot, the kernel regenerates `/proc` and `/sys`, udev recreates device nodes, and the system starts fresh with all your installed software and configuration intact.

White-out Files

When you delete a file that exists in a read-only SFS layer, the overlay filesystem creates a "white-out" marker that hides the original file. During remaster, EERemaster handles these correctly – the deleted files are excluded from the new SFS, so your deletions become permanent. If you uninstalled a package that came with the base system, it stays uninstalled in the remastered image.

Typical Workflow

1. Boot EEOS and set up the system exactly how you want the final image – install software, configure settings, customize the desktop, set up HHFE, etc.
2. Remove any personal files, credentials, browser history, or other data you don't want baked into the image.
3. Run EEBackup to create a snapshot (safety net in case the remaster has issues).
4. Open EERemaster.
5. On the checklist: leave 2 and 4 on. Turn on 5 if you want an ISO. Turn on 3 if deploying to different hardware.
6. Select a working partition with enough free space.
7. Wait for the build to complete.
8. The new SFS (and ISO, if selected) appears on the working partition.
9. Test the image by booting from it on a different machine before deploying widely.

File Paths

File/Path	Purpose
`/usr/sbin/eeos-remaster-classic`	Classic remaster script (380 lines, sh/Xdialog)
`/usr/sbin/eeos-remaster-alternative`	Alternative method (378 lines)
`/usr/local/bin/eeos-remaster`	Wrapper script (15 lines)
`/etc/rc.d/EEOSSTATE`	Boot mode detection
`/etc/DISTRO_SPECS`	Distribution version and naming info

Troubleshooting

"Not enough space" on the working partition: - The build needs roughly 1.5x the expected SFS size in free space. A heavily customized system with lots of installed packages could produce an 800MB+ SFS, needing 1.2GB+ free. - Choose a different partition with more room, or free space on the current one. - Don't delete files in a panic – use `df -h` to assess the situation and make informed decisions.

Build takes a very long time: - Squashfs compression is CPU-intensive. On older or low-power machines, a full remaster can take 15-30 minutes. The rxvt progress window shows what `mksquashfs` is doing. - If it seems stuck, check that the working partition hasn't run out of space. A silent ENOSPC during compression can look like a hang.

Remastered image won't boot: - The most common cause is a missing kernel or initrd. Verify the ISO (if you made one) contains `vmlinuz` and `initrd.gz` by mounting it and checking. - If the SFS boots but hardware doesn't work, you may have excluded firmware. Step 2 of the build process should have preserved it, but verify by checking `/lib/firmware/` in the mounted SFS. - If you used option 3 (discard hardware settings) but the target machine is the same as the build machine, hardware auto-detection should reconfigure on first boot. If it doesn't, boot with the "Maximum Compatibility" entry.

SFS file is unexpectedly large: - You may have data in unexpected locations under the root filesystem. Check for large files in `/root`, `/usr/local`, or `/opt` that you don't need. - Loaded extra SFS modules are included by default. If you loaded a large module (like a development environment) that you don't want in the final image, use option 1 to exclude it.

Image is missing packages or config that should be there: - Verify the packages were installed in the overlay, not in a separate SFS module that got excluded. - Check that the config files live in `/etc` and not in a path that's on the exclusion list. - If running from a save folder or save file, make sure you weren't in RAM-only mode when you installed the packages – RAM-only changes don't persist to the overlay.

ISO file is much larger than the SFS: - The ISO includes the kernel, initrd, and boot loader in addition to the SFS. The overhead is typically 10-30MB. If it's significantly larger, check that extra files didn't end up in the ISO build directory.

EEStick Installers

Binaries: `/usr/sbin/eeos-stick-installer`, `/usr/sbin/eeos-e4-stick-installer`, `/usr/sbin/eeos-e3-stick-installer`, `/usr/sbin/eeos-f2-stick-installer` **Desktop entries:** `eeos-stick-installer.desktop`, `eeos-e4-stick-installer.desktop`, `eeos-e3-stick-installer.desktop`, `eeos-f2-stick-installer.desktop`

Purpose

Write EEOS to a USB stick with a specific target filesystem. Four variants for different use cases:

Installer	Target FS	Best For
EEStick (generic)	auto-detect	general use
EE4Stick	ext4	persistent saves, most compatible
EE3Stick	ext3	legacy compatibility
EEF2Stick	f2fs	flash-optimized, lower write amplification

When to Use Which

- **ext4:** Default choice. Works everywhere, good for persistent saves, journaling protects against corruption on unclean shutdown.
- **f2fs:** Best for USB sticks that will see heavy write activity. F2FS is designed for flash storage and reduces write amplification, extending the USB stick's lifespan.
- **ext3:** Only if you need compatibility with very old systems that can't read ext4.

Part 6: Utilities

EEArchive

Binary: `/usr/local/bin/eeos-archive` **Desktop entry:** `eeos-archive.desktop` **Menu location:** Utilities > EEArchive

Purpose

Compression and extraction tool. Handles all common archive formats.

Supported Formats

Format	Compress	Extract
tar	yes	yes
tar.gz / .tgz	yes	yes
tar.xz	yes	yes
tar.bz2	yes	yes
zip	yes	yes
7z	yes	yes
rar	no	yes (extract only)
deb	no	yes
rpm	no	yes
cpio	yes	yes
ee (.pet)	no	yes
sfs	no	yes (unsquashfs)

Usage

- **GUI:** Right-click any archive in the file manager to open with EEArchive. Or select files, right-click, and choose "Compress" to create an archive.
- **CLI:** `eeos-archive <file>` to extract, or use standard tools (`tar`, `7z`, `zip`, etc.) directly.

EEAdBlocker

System-level ad, tracker, and malware domain blocking.

Source: `/usr/sbin/eeos-advert-blocker` (160 lines, bash) **Desktop entry:** `eeos-advert-blocker.desktop` **Menu location:** Utilities > EEAdBlocker

What It Does

Blocks advertising, tracking, malware, and telemetry domains at the system level by manipulating `/etc/hosts`. Every application on the system – browsers, desktop apps, background services, anything that resolves a domain name – goes through the hosts file before hitting external DNS. Blocked domains get redirected to `0.0.0.0` (null route) and never resolve.

This isn't a browser extension. It doesn't depend on a proxy. It works for everything, including things you can't install extensions into.

How It Works

The script downloads curated blocklists of known bad domains – ad servers, tracking pixels, malware command-and-control endpoints, telemetry collectors – and merges them into `/etc/hosts`. Each entry follows the format:

```
0.0.0.0 ads.example.com
0.0.0.0 tracker.example.net
0.0.0.0 telemetry.bigcorp.io
```

When any application tries to resolve one of these domains, the system returns `0.0.0.0` instead of the real IP. The connection fails silently. No data leaves the machine.

Operations

The script presents a dialog with three options:

Enable / Update blocking:

1. Downloads the latest blocklist from curated upstream sources
2. Backs up the current `/etc/hosts` to a timestamped copy
3. Merges the blocklist entries into the hosts file, preserving any custom entries you've added manually
4. Activates blocking immediately

The blocklist sources maintain lists of ad servers, tracking pixels, malware distribution domains, cryptominer scripts, and OS telemetry endpoints. Updates pull the latest version of these lists.

Disable blocking:

Restores the original `/etc/hosts` from the backup copy, stripping all blocklist entries. Network access to previously blocked domains is restored immediately for new connections. Existing connections that were blocked remain blocked until the application retries.

Check status:

Shows whether blocking is currently active and how many domains are in the blocklist. Useful for confirming the block is in place after a reboot or session restore.

Technical Details

- Original `/etc/hosts` is backed up before any modification. The backup is what gets restored when you disable blocking.
- No background daemon or service runs. The hosts file is static – once written, it works without any process keeping it alive.
- Changes take effect immediately for new DNS lookups. Already-established connections to blocked domains aren't terminated (the app would need to reconnect).
- The hosts file survives reboots if the system uses persistent session storage. On a stateless boot without persistence, you'd need to re-enable after each reboot.
- Works offline once enabled. The blocklist is stored locally in the hosts file itself. You only need internet to download/update the list.
- Does not touch system DNS configuration (`/etc/resolv.conf`), only the hosts file.

Why This Matters for Centers

Center machines running HHFE don't need ads, tracking, or telemetry phoning home. Enabling EEAdBlocker on center deployments:

- Reduces network traffic and speeds up page loads (fewer requests to ad servers)
- Prevents tracking of center machines by analytics companies
- Blocks known malware distribution domains as a defense-in-depth layer
- Zero maintenance once enabled – the blocklist doesn't expire
- No per-browser configuration needed

For center deployments built with EERemaster, enable EEAdBlocker before remastering so every machine gets the blocklist baked in.

EEChecksum

Graphical file integrity verification.

Source: `/usr/sbin/eeos-md5sum.sh` (296 lines, bash/gtkdialog) **Desktop entries:** `eeos-md5sum.desktop` , `eeos_md5sum.desktop` **Menu location:** Utilities > EEChecksum

What It Does

Generates and verifies cryptographic checksums for any file. Supports MD5, SHA1, SHA256, and SHA512. Tells you whether a file is exactly what it should be, or whether something changed – corruption during download, a bad sector on disk, or tampering.

Use Cases

- Verify a downloaded ISO image against the published checksum before writing it to a USB stick
- Verify downloaded packages or updates before installation
- Generate checksums for files you're distributing to others
- Detect corrupted files after copying to/from USB drives or across a network

- Confirm two copies of a file are identical without doing a byte-by-byte diff

The Interface

The application opens a gtkdialog window with:

1. **File selection:** Browse button to pick the file you want to check. Or type/paste the path directly.
2. **Hash algorithm selector:** Radio buttons for MD5, SHA1, SHA256, SHA512.
3. **Expected hash field:** A text input where you paste the known-good hash value (from a download page, release notes, or `.sha256sum` file).
4. **Generate button:** Computes the hash of the selected file and displays it. Use this when you want to produce a hash to share or record.
5. **Verify button:** Computes the hash and compares it character-by-character against whatever you pasted in the expected hash field.
6. **Result display:** Shows **MATCH** (green) or **MISMATCH** (red). No ambiguity.

Hash Generation Mode

Select a file, pick an algorithm, and click Generate without entering an expected hash. The tool computes the hash and displays it. You can copy this string to:

- Include in release notes alongside a downloadable file
- Save to a `.sha256sum` or `.md5sum` file
- Compare manually against another source
- Store for later verification

Hash Verification Mode

Select a file, paste the expected hash into the field, pick the matching algorithm, and click Verify. The tool computes the file's actual hash and compares it against the expected value. Any difference – even a single character – means the file is corrupted, incomplete, or has been modified since the hash was generated.

A mismatch after downloading an ISO means: don't write it to a USB stick. Re-download it. If it mismatches again, try a different mirror or check that the published checksum is actually for the file you downloaded (version numbers, architecture, etc.).

Which Algorithm to Use

Algorithm	Speed	Security	When to Use
MD5	Fastest	Cryptographically broken	Fine for detecting accidental corruption. Not safe against intentional tampering.
SHA1	Fast	Weak (collision attacks exist)	Legacy use only. Some older tools still publish SHA1 hashes.
SHA256	Moderate	Current standard	

Algorithm	Speed	Security	When to Use
			Use this for anything important. Most download pages publish SHA256.
SHA512	Slowest	Maximum	Critical system images, forensic verification.

Match whatever the source provides. If a download page gives you a SHA256 hash, verify with SHA256. If it gives you MD5, use MD5 – but be aware it only proves the file wasn't accidentally corrupted, not that it wasn't deliberately altered.

Command-Line Equivalents

The GUI wraps standard Linux utilities. For scripting or terminal use:

```
# Generate
md5sum /path/to/file
sha1sum /path/to/file
sha256sum /path/to/file
sha512sum /path/to/file

# Verify against a checksum file
sha256sum -c checksums.sha256
```

Troubleshooting

Hash doesn't match but the file seems fine:

Check that you're using the right algorithm. A SHA256 hash compared against an MD5 will never match. Also check that you copied the full hash string – partial copies are a common mistake.

File takes a long time to hash:

Normal for large files (ISOs, disk images). SHA512 on a 4GB file over USB 2.0 can take several minutes. The interface may appear frozen – it isn't. Wait for it to complete.

Can't find the expected hash:

For EEOS ISOs, check the release notes or the download page. For other software, look for a file named `SHA256SUMS`, `checksums.txt`, or similar alongside the download link.

EEPDF

Lightweight PDF conversion utility.

Source: `/usr/sbin/eeos-pdf` (248 lines, bash) **Desktop entry:** `eeos-pdf-convert.desktop` **Menu location:** Utilities > EEPDF (or Office > PDF Converter)

What It Does

Converts documents, images, and text files to PDF format. Not a full office suite – it's the quick-and-dirty converter for when you need a PDF and don't want to open LibreOffice.

Supported Input Formats

Format	Extensions	Notes
Plain text	`.txt`	Wrapped and formatted as PDF pages with monospace font
HTML	`.html`, `.htm`	Rendered to PDF preserving layout and basic CSS
Images	`.jpg`, `.jpeg`, `.png`, `.bmp`, `.tiff`, `.gif`	Each image becomes a full PDF page
PostScript	`.ps`, `.eps`	Converted via Ghostscript
Rich Text	`.rtf`	Basic formatting preserved (bold, italic, fonts)
Multiple files	mixed	Batch mode – combine multiple inputs into one PDF, or convert each individually

How to Use

1. Open EEPDF from the menu (Utilities > EEPDF, or Office > PDF Converter)
2. Select one or more input files via the file browser
3. Choose the output filename and location
4. Click Convert
5. The PDF appears at the specified location

For batch conversion of multiple files, select them all in the file browser. You'll get the option to combine them into a single PDF or create individual PDFs for each input file.

Technical Backend

Under the hood, EEPDF dispatches to whichever conversion tool fits the input format:

- **PostScript/EPS:** Ghostscript (`gs`) with PDF output profile
- **Images:** ImageMagick (`convert`) or similar rasterization pipeline
- **HTML:** The system's HTML rendering engine
- **Text:** Formatted through an intermediate stage, then to PDF via Ghostscript
- **RTF:** Converted through the available RTF-to-PostScript pipeline, then to PDF

No external cloud services. Everything runs locally.

Tips

- For office documents (DOC, DOCX, ODT, XLS, XLSX), use LibreOffice's File > Export as PDF instead. EEPDF doesn't handle the Office XML formats.
- For web pages with complex JavaScript-rendered content, the browser's own "Print to PDF" will give better results than feeding the HTML to EEPDF.
- EEPDF is best for quick conversions of simple stuff – a text log to PDF for emailing, a folder of screenshots into a single PDF document, a PostScript file from an old printer queue.
- Image-to-PDF preserves the original image resolution. If you need a specific page size, resize the images first.

Troubleshooting

Conversion fails silently (no output file created):

Check that the required backend tools are installed. Run `which gs` and `which convert` from a terminal. If either is missing, install through EESoftware or EEPackages.

HTML conversion looks wrong:

Complex HTML with modern CSS or JavaScript won't render perfectly. EEPDF uses a basic HTML renderer, not a full browser engine. For complex pages, use the browser's Print to PDF.

Text PDF has weird line breaks:

The text wrapper uses a fixed character width. Very long lines without whitespace (like log files with long paths) may break awkwardly. Pre-format the text file or use a monospace-friendly line length (80-120 characters).

EEHelp

Binary: `/usr/sbin/eeos-help` **Desktop entry:** `eeos-help.desktop` **Menu location:** Help > EEHelp

Purpose

Opens the local EEOS documentation in the default web browser. All content is stored locally on the USB – no internet required.

Documentation Location

Help files are stored at: - `/usr/share/doc/` – package documentation - `/usr/share/eeos/` – EEOS-specific reference files (EEAI knowledge base, software reference, complete manual) - `/usr/share/doc/eeos-manual/` – the full EEOS manual

---

EENetwork Shares

Binary: `/usr/local/bin/eeos-network-shares` **Desktop entry:** `eeos-network-shares.desktop` **Menu location:** Network > EENetwork Shares

Purpose

Samba (SMB/CIFS) file sharing configuration. Set up the machine to share files on the local network or access shares from other machines (Windows, Mac, Linux).

Operations

- **Browse network:** Discover and access shares from other machines on the local network
- **Create share:** Share a local folder with other machines
- **Mount remote share:** Mount a network share as a local directory
- **Configure Samba:** Edit the Samba server configuration for persistent shares

Configuration

Samba configuration: `/etc/samba/smb.conf` Init script: `/etc/init.d/rc.samba`

EEUtilities

Binary: `/usr/sbin/eeos-utility-suite` **Desktop entry:** `eeos-utility-suite.desktop` **Menu location:** Utilities > EEUtilities

Purpose

Collection of system maintenance utilities in a single interface: - Disk usage analyzer (gdmap) - File search (pfind/fsearch) - Checksum verification (EEChecksum) - Disk partitioning links (GParted) - Package conversion tools (dir2deb, dir2pet, dir2sfs, dir2targz) - ISO creation (dir-to-iso) - Multi-file rename - Dependency listing

---

Part 7: Remote Management

EERemote

Agent binary: `/usr/local/bin/eeos-remote-agent` **Control binary:** `/usr/local/bin/eeos-remote-ctl` **Config file:** `/etc/eeos/remote-agent.conf` **Service file:** `/usr/lib/systemd/system/eeos-remote-agent.service` (if systemd is used)

Purpose

Encrypted remote management for EEOS center deployments over Tailscale VPN. Allows a management machine to check status, restart services, push configurations, pull diagnostics, and trigger reboots on remote center machines.

Architecture

Two components:

1. **EERemote Agent** (`eeos-remote-agent`): Runs as a listener on each center machine. Waits for signed commands from authorized control instances.

2. **EERemote Control** (`eeos-remote-ctl`): CLI tool on the management machine. Sends signed commands to remote agents.

Security Model

- **Transport encryption:** All traffic goes through Tailscale/WireGuard (encrypted tunnel)
- **Command authentication:** HMAC-SHA256 signed commands using a pre-shared key
- **Nonce protection:** Destructive operations (reboot, config push) require a unique nonce to prevent replay attacks
- **Logging:** Every received command is logged to `/var/log/eeos-remote.log`

Configuration (`/etc/eeos/remote-agent.conf`)

```
# pre-shared key for HMAC signing (must match on both agent and control)
PSK=<shared-secret>

# listen address (Tailscale IP only)
LISTEN_ADDR=0.0.0.0

# listen port
LISTEN_PORT=9847

# allowed commands
ALLOWED_COMMANDS=status,restart-hhfe,restart-network,push-config,pull-diag,reboot

# log file
LOG_FILE=/var/log/eeos-remote.log
```

Commands

Command	Description	Destructive?
`status`	Return system status (CPU, RAM, HHFE state, network)	No
`restart-hhfe`	Kill and restart DOSBox/HHFE	No
`restart-network`	Restart EENetwork service	No
`push-config`	Push a configuration file to the remote machine	Yes (nonce required)
`pull-diag`	Request a diagnostic report from the remote machine	No
`reboot`	Reboot the remote machine	Yes (nonce required)

Setup

1. Both machines need Tailscale installed and logged in to the same tailnet
2. Generate a shared key on one machine: `openssl rand -hex 32`
3. Set the key in `/etc/eeos/remote-agent.conf` on both machines
4. Start the agent: `eeos-remote-agent --enable`
5. Test: `eeos-remote-ctl --target <tailscale-ip> status`

Troubleshooting

Commands not reaching the center machine: 1. Check Tailscale status on both ends: `tailscale status` 2. Verify the agent is running: `pgrep -f eeos-remote-agent` 3. Check the firewall isn't blocking port 9847: `iptables -L | grep 9847`

HMAC verification fails: The pre-shared keys don't match between the agent and control instances. Re-copy the key from one machine to the other. Check for trailing whitespace or newlines in the config file.

Command log: All commands are logged to `/var/log/eeos-remote.log` with timestamp, source IP, command, and result. Review this for debugging.

Part 8: Kiosk and Lockdown

EEKiosk

Binary: `/usr/local/bin/eeos-kiosk` **Init script:** `/usr/local/bin/eeos-kiosk-init` **Status display:** `/usr/local/bin/eeos-kiosk-status` **Error screen:** `/usr/local/bin/eeos-kiosk-error-screen` **Watchdog:** `/usr/local/bin/hhfe-watchdog` **Config file:** `/etc/eeos/kiosk.conf` **EEDesktop kiosk config:** `labwc-kiosk-rc.xml`, `labwc-kiosk-autostart`, `labwc-kiosk-themerc` **EETaskbar kiosk config:** `sfwbar-kiosk.config`

Purpose

HHFE lockdown mode for unattended center operation. Disables desktop access (menu, file manager, terminal) and runs HHFE fullscreen with automatic crash recovery.

Architecture

EEKiosk replaces the standard desktop session with a locked-down environment:

1. **Kiosk init** (`eeos-kiosk-init`): Runs at session start, replaces the normal desktop autostart
2. **EEDesktop kiosk mode:** Custom compositor config (`labwc-kiosk-rc.xml`) that removes window decorations, disables window movement, and blocks keyboard shortcuts
3. **EETaskbar kiosk mode:** Minimal status bar showing only essential information (time, network status, kiosk lock indicator)
4. **HHFE fullscreen:** DOSBox runs maximized with no window chrome
5. **Watchdog:** Monitors HHFE and restarts it if it crashes

What Gets Locked

- Desktop right-click menu: disabled
- File manager: not started
- Terminal access: blocked
- Alt+Tab: disabled
- Window decorations: removed
- Desktop icons: hidden
- System tray: minimal (status bar only)

Watchdog (`hhfe-watchdog`)

The watchdog is a background process that monitors the DOSBox/HHFE process:

1. Checks if DOSBox is running every 10 seconds
2. If DOSBox has died:
   - Waits `backoff` seconds (starts at 5, doubles each failure, caps at 300)
   - Attempts to restart HHFE
   - If restart succeeds, resets the backoff timer
   - If restart fails, shows the error screen and continues trying
3. If DOSBox has been running for more than 60 seconds since last restart, resets the failure counter

The exponential backoff prevents rapid restart loops if HHFE has a persistent problem (e.g., missing INVOKE.TXT, corrupted config).

Error Screen (`eeos-kiosk-error-screen`)

Displayed when the watchdog can't restart HHFE. Shows: - "HHFE is not responding" message - Basic diagnostic info (last restart attempt, failure count) - Instructions to contact support - The maintenance unlock key combination

Unlocking

1. Press the maintenance key combination (configurable, default: Ctrl+Alt+Shift+M)
2. Enter the maintenance PIN
3. Desktop unlocks to full access
4. Reboot to restore kiosk mode

PIN Management

- Set PIN: `eeos-kiosk --set-pin`
- PIN is stored as SHA-512 hash in `/etc/eeos/kiosk.conf`
- Default PIN must be changed before deploying to a center

Configuration (`/etc/eeos/kiosk.conf`)

```
# kiosk mode enabled
EEOS_KIOSK_ENABLED=yes

# maintenance PIN (SHA-512 hash)
EEOS_KIOSK_PIN_HASH=<sha512-hash>

# show status bar
EEOS_KIOSK_STATUSBAR=yes

# watchdog settings
EEOS_KIOSK_WATCHDOG=yes
EEOS_KIOSK_BACKOFF_INITIAL=5
EEOS_KIOSK_BACKOFF_MAX=300

# unlock key combination
EEOS_KIOSK_UNLOCK_KEYS=ctrl+alt+shift+m
```

Troubleshooting

Kiosk won't unlock: Boot in Standard mode from the GRUB menu (not the kiosk entry). This bypasses kiosk entirely and gives you full desktop access.

Watchdog keeps restarting HHFE but it keeps crashing: The problem is with HHFE/DOSBox, not kiosk mode. Check: 1. Run EEDiagnostics from a terminal (unlock kiosk first or boot in Standard mode) 2. Check that INVOKE.TXT exists at `/opt/hhfe/INVOKE.TXT` 3. Check DOSBox config at `/opt/hhfe/.dosbox/dosbox-hhfe.conf` 4. Check `/var/log/eeos-hhfe.log` for error details

Status bar missing: Check `EEOS_KIOSK_STATUSBAR=yes` in `/etc/eeos/kiosk.conf`. If yes and still missing, verify the EETaskbar is running: `pgrep sfwbar`.

Kiosk mode not activating on boot: Ensure the GRUB boot entry includes the kiosk flag in the kernel command line. The kiosk init script checks for this flag before activating lockdown.

Part 9: Biometrics

EEVitals

Module directory: `v2/vitals/` **Knowledge base:** `/usr/share/eeos/EEAI_VITALS_KNOWLEDGEBASE.md` **Compliance framework:** `v2/vitals/docs/COMPLIANCE_FRAMEWORK.md` **Center operator guide:** `v2/vitals/docs/CENTER_OPERATOR_GUIDE.md` **Marketing guidelines:** `v2/vitals/docs/MARKETING_GUIDELINES.md` **Session report template:** `v2/vitals/docs/SESSION_REPORT_TEMPLATE.md` **Quick reference card:** `v2/vitals/docs/QUICK_REFERENCE_CARD.md`

Purpose

Biofield Vitality Index (BVI) measurement platform. Records heart rate variability (HRV) data from a BLE chest strap during EESystem sessions and computes a composite wellness score.

The BVI Score (0-100)

Three weighted domains:

AUTONOMIC RECOVERY (50% OF SCORE)

Derived from HRV metrics captured via BLE heart rate sensor. Raw RR intervals (time between heartbeats in ms) are processed into:

RMSSD (Root Mean Square of Successive Differences) - Measures parasympathetic vagal tone - Higher values = rest-and-recover branch more active - Single strongest real-time indicator of relaxation depth - Client display name: "Relaxation Depth"

SDNN (Standard Deviation of NN intervals) - Measures total autonomic variability (both branches) - Higher values = more adaptable, resilient nervous system - Client display name: "Heart Rhythm Variability"

LF Power (Low Frequency, 0.04-0.15 Hz) - Baroreflex modulation and mixed autonomic activity - NOT "stress" despite what consumer apps claim - Reflects active cardiovascular regulation engagement - Client display name: part of "Active Balance"

HF Power (High Frequency, 0.15-0.40 Hz) - Purest measure of parasympathetic activation - Linked to respiratory cycle - High values = deep recovery mode - Client display name: "Rest Response"

DFA alpha-1 (Detrended Fluctuation Analysis) - Fractal complexity of heart rhythm regulation - Value near 1.0 at rest = healthy fractal behavior - Below 0.75 = system under load - Above 1.3 = rigid, over-correlated regulation - Requires 2+ minutes of continuous data (128+ beats) - Not displayed to clients directly; contributes to domain score

Normalization: lnRMSSD (natural log, same as WHOOP/Oura/Elite HRV) and log-normal for spectral power.

SUBJECTIVE VITALITY (25% OF SCORE)

Captured in a 25-second pre-session check-in: - **VAS Energy slider:** 0-100 rating of current energy level - **Two WHO-5 adapted items:** "I have felt cheerful and in good spirits" and "I have felt calm and relaxed" (each rated 0-5)

This exists because biometric data alone can't capture how someone feels. Excellent HRV + feeling terrible = the biometrics miss something. The subjective layer grounds the score.

SESSION RESPONSE (25% OF SCORE)

Delta between "before" and "during" phases: - RMSSD delta: parasympathetic tone increase during session - HF power delta: rest response deepening - Heart rate delta: how much HR slowed (lower during = more relaxed)

This is the unique session-based measurement. Not just "where are you now" but "how much did your body shift during this specific session."

Score Interpretation

Score	Status	What to tell the client
85-100	Thriving	"Your autonomic flexibility was very high during this session with a strong relaxation response across all metrics."
70-84	Energized	"Your nervous system showed a solid shift into recovery mode during this session."
50-69	Baseline	"Your body showed a moderate response during this session. Responses vary between sessions and this is within normal range."
30-49	Depleted	"Your autonomic metrics suggest your nervous system may be carrying some recovery debt. Rest and recovery activities may help."
Below 30	Rest	"Your metrics suggest prioritizing rest. If this pattern persists across multiple sessions consider discussing with a wellness practitioner."

Personal Baseline System

Scores are normalized against the individual's rolling history, not population averages: - **Days 1-7:** Calibration period. Raw metrics shown, no BVI score computed. - **Days 8-14:** Provisional BVI using 50/50 blend of personal and population reference. - **Day 15+:** Full personal baseline with 60-day rolling window.

A BVI of 72 means "72% of your personal vitality capacity right now," not "72nd percentile of all humans."

Session Phases

Phase	Duration	Purpose
Pre	First 5 minutes	Baseline before HHFE starts. Client is seated, sensor on.
During	Main session (30 min to 3 hours)	HHFE is running. Rolling 5-minute BVI windows.
Post	Last 5 minutes	Carryover after HHFE stops.

The before/during/after comparison is the primary output on the session report.

Hardware Requirements

Any BLE device implementing the Heart Rate Profile (UUID 0x180D): - **Polar H10, H9, OH1** – gold standard for HRV accuracy - **Garmin HRM-Pro, HRM-Dual** - **Wahoo TICKR** - **Most gym-grade BLE chest straps**

The sensor streams RR intervals at 1/1024 second resolution via BLE notifications. Multiple RR values can arrive per packet (up to 9 at high heart rates).

Wrist-based devices (Apple Watch, Fitbit, Garmin watches) do NOT work. They don't expose RR intervals via BLE.

Data Export

Each session produces: - `sess_YYYYMMDD_HHMMSS_rr.csv` – Kubios-compatible RR interval time series (seconds) - `sess_YYYYMMDD_HHMMSS_metadata.json` – full metadata with all computed metrics and BVI scores per phase

Stored on the encrypted partition. Exportable to USB or email for researchers.

Setup Process

1. Client puts on BLE heart rate chest strap (wet contact sensors for conductivity)
2. Open EEVitals
3. Sensor appears automatically via BLE scan
4. Client completes 25-second check-in (energy slider + two wellness questions)
5. Start session. EEVitals records continuously.
6. After session ends, report is generated automatically.

Compliance

EEVitals is a wellness measurement tool. It does **not**: - Diagnose medical conditions - Recommend treatments - Make prescriptive health claims - Replace medical devices or professional medical advice

All client-facing language must use the vocabulary from the Compliance Framework. EEAI enforces this in all responses.

Troubleshooting

Sensor doesn't appear: 1. Check Bluetooth is enabled on the EEOS machine 2. Wet the chest strap contact pads (dry skin = no signal) 3. Ensure the strap is in contact with skin (snug, not loose) 4. Check battery in the sensor module 5. Try `bluetoothctl scan on` from a terminal to see if the sensor broadcasts

HRV values erratic in the first minute: Normal. The system needs ~60 seconds of clean data for algorithms to stabilize. The first few RR intervals often have motion artifacts from putting the strap on.

DFA alpha-1 shows as unavailable: Not enough continuous data yet. DFA needs 128+ consecutive clean beats (about 2+ minutes). Wait longer or check for signal dropouts.

BVI score not appearing: In the calibration period (days 1-7), BVI is not computed. Raw metrics are shown instead. After day 7, provisional scoring begins.

Large score variations between sessions: This is expected and normal. HRV is highly sensitive to: - Sleep quality the night before - Caffeine intake - Stress levels - Hydration - Time of day - Exercise in the prior 24 hours

Trends over multiple sessions are more meaningful than any single session score.

Part 10: Boot System

EEAI Boot Profile Selector

Location: Built into the init script (`eeos-init/init`) **Trigger:** Runs during init, before the desktop loads **Boot menu:** Configured in `branding/boot/menu.lst`

Purpose

Smart boot profile auto-detection that runs during the init process. Detects CPU, RAM, and GPU hardware, then suggests the optimal boot configuration for the detected hardware.

Hardware Detection and Recommendations

Hardware	Recommendation
8GB+ RAM	"Copy to RAM" (entire OS loads into memory, USB can be removed after boot)
2-8GB RAM	"Standard desktop" (run from USB with overlay)
Under 2GB RAM	"CLI only" (text console, no desktop – not enough RAM for GUI)

Boot Countdown

Shows a live countdown (default 10 seconds) with the auto-detected recommendation. The user can: - Let it count down for automatic selection - Press a key to override and choose manually - Press a specific number to jump to that boot profile

GRUB Boot Entries

The full boot menu configured in `menu.lst` :

Entry	Description	Kernel params
EEAI Assistant	Smart profile selection (described above)	`eeai`
EEOS 1.0	Standard desktop boot	(default)
Copy to RAM	Loads entire OS into memory	`eeos=copy`
Keep on USB	Run from USB, don't copy	(default, no copy)
No X (CLI only)	Text console, no desktop	`eeos=nox`

Entry	Description	Kernel params
Recovery Shell	Drops to a root shell	`eeos=rdsh`
RAM only (ephemeral)	No persistence, all changes lost on shutdown	`eeos=ram`
Safe Video	Forces nomodeset for GPU issues	`nomodeset`
Intel Mac / Broadwell+	Compatibility mode for Intel Macs	`nomodeset i915.modeset=0`
NVIDIA Safe	Disables nouveau for NVIDIA issues	`modprobe.blacklist=nouveau`
Maximum Compatibility	Disables everything that might cause problems	`nomodeset noapic noacpi nosplash`

When to Use Each Entry

Normal operation: EEAI Assistant or EEOS 1.0 **Center deployment with enough RAM:** Copy to RAM (fastest performance, USB can be removed) **Troubleshooting display issues:** Safe Video first, then NVIDIA Safe, then Maximum Compatibility **Emergency access:** Recovery Shell (drops to root prompt for manual fixes) **Testing without persistence:** RAM only (nothing saved, clean slate on reboot)

Part 11: Additional EE Components

EEMode

Binary: `/usr/local/bin/eeos-mode`

Purpose

Queries or sets the current EEOS operating mode. Reads from `/etc/rc.d/EEOSSTATE` which tracks the boot mode, session state, and system configuration flags.

EELock

Binary: `/usr/sbin/eeos-lock`

Purpose

Screen lock utility. Activates the screen lock (swaylock under Wayland, or xscreensaver/vlock under X11) to prevent unauthorized access when the operator steps away.

Part 12: EEOS Desktop Environment

EEDesktop Compositor

Binary: `/usr/bin/labwc` **Config directory:** `~/.config/labwc/` **Session file:** `/usr/share/wayland-sessions/labwc.desktop` **Autostart:** `~/.config/labwc/autostart`

What It Is

The EEOS window compositor handles window management, keyboard and mouse input, display output, and visual compositing. It's a Wayland-native stacking compositor.

Configuration uses standard XML syntax familiar to Linux desktop administrators.

Configuration Files

All config files live in `~/.config/labwc/`:

File	Purpose
`rc.xml`	Main config - keybinds, mouse binds, window rules, focus, themes, workspaces
`menu.xml`	Right-click context menus and root menu
`autostart`	Shell script that runs after the compositor starts - launches panels, background, etc.
`shutdown`	Shell script that runs when session ends
`environment`	Environment variables set before autostart runs
`themerc`	Window decoration theme (title bar colors, fonts, borders)

To reload config without restarting: `labwc --reconfigure` or send SIGHUP.

Key Configuration Sections (rc.xml)

CORE SETTINGS

```
<core>
  <decoration>server</decoration>    <!-- server-side decorations (titlebars) -->
  <gap>0</gap>                        <!-- gap between tiled windows -->
  <adaptiveSync>no</adaptiveSync>     <!-- VRR/FreeSync -->
</core>
```

WINDOW PLACEMENT

```
<placement>
  <policy>cascade</policy>    <!-- center | automatic | cursor | cascade -->
  <cascadeOffset x="40" y="30" />
</placement>
```

FOCUS BEHAVIOR

```
<focus>
  <followMouse>no</followMouse>          <!-- focus follows mouse pointer -->
  <raiseOnFocus>no</raiseOnFocus>        <!-- auto-raise focused windows -->
</focus>
```

KEYBOARD BINDINGS

Modifiers: `S` (Shift), `C` (Control), `A` (Alt), `W` (Super/Win key)

Default bindings in EEOS: | Keys | Action | |–|–| | A-Tab | Next window | | A-S-Tab | Previous window | | W-Return | Open terminal | | A-F4 | Close window | | W-a | Toggle maximize | | W-Left/Right | Snap to left/right half | | W-Up/Down | Snap to top/bottom half | | A-Space | Window menu |

Custom bindings are added in `<keyboard>` section:

```
<keybind key="W-Return">
  <action name="Execute" command="lxterminal" />
</keybind>
```

MOUSE BINDINGS

Contexts: TitleBar, Client, Frame, Desktop, Root, Border, and all corner/edge regions.

Actions: Press, Release, Click, DoubleClick, Drag, Scroll.

Buttons: Left, Middle, Right, Side, Extra, Forward, Back.

WINDOW SNAPPING

```
<snapping>
  <range>
    <inner>10</inner>
    <outer>10</outer>
  </range>
  <topMaximize>yes</topMaximize>
</snapping>
```

WINDOW RULES

Match windows by title, app identifier, or type and apply actions:

```
<windowRules>
  <windowRule identifier="dosbox" skipTaskbar="no" fixedPosition="no">
    <action name="MaximizeWindow" />
  </windowRule>
</windowRules>
```

THEME CONFIGURATION

```
<theme>
  <name>EEOSFlat</name>
  <cornerRadius>8</cornerRadius>
  <titlebar>
    <layout>icon:iconify,max,close</layout>
    <showTitle>yes</showTitle>
  </titlebar>
  <font place="ActiveWindow" name="sans" size="10" />
</theme>
```

WORKSPACES

```
<desktops number="1">
  <names><name>Desktop</name></names>
  <popupTime>1000</popupTime>
</desktops>
```

INPUT DEVICE CONFIGURATION (LIBINPUT)

```
<libinput>
  <device category="touchpad">
    <naturalScroll>no</naturalScroll>
    <tap>yes</tap>
    <pointerSpeed>0.0</pointerSpeed>
    <accelProfile>adaptive</accelProfile>
    <disableWhileTyping>yes</disableWhileTyping>
  </device>
</libinput>
```

Autostart

The `~/.config/labwc/autostart` file runs after the compositor initializes. In EEOS, it typically starts: - EETaskbar (status bar) - swaybg (wallpaper) - swayidle (screen blanking) - conky (system monitor widget) - HHFE autostart - Network monitor applet

Troubleshooting

Desktop doesn't start: Check `~/.config/labwc/autostart` for errors. Try running `labwc` from a TTY console to see error output.

Window decorations missing: Check `<core><decoration>server</decoration></core>` in rc.xml. Some apps request client-side decorations; use window rules to override.

Keyboard shortcuts not working: Check `<keyboard>` section in rc.xml. Run `labwc --reconfigure` after changes.

Display configuration: Use `wlr-randr` to list and configure outputs. For a GUI, use `wdisplays`.

EETaskbar

Binary: `/usr/bin/sfwbar` **Config directory:** `~/.config/sfwbar/` **System configs:** `/usr/share/sfwbar/` **Autostart:** Started from EEDesktop autostart

What It Is

The EEOS taskbar and status bar for Wayland desktop sessions. Provides the bottom panel with the application menu, task list, system tray, clock, and status indicators.

Configuration

Reads `sfwbar.config` from `~/.config/sfwbar/`, falling back to `/usr/share/sfwbar/`. Config files use the Api2 format (indicated by `#Api2` tag).

Widget Types

Widget	Purpose
taskbar	Shows running application windows
pager	Workspace switcher
tray	System tray (status notifier items)
grid	Layout container for organizing other widgets
label	Text display with Pango markup
scale	Progress bar
chart	Time-series graph
image/button	Icon with click handler
entry	Text input field

Layout System

Widgets are positioned using `loc(x,y[,w,h])` within grid containers. Bars can be placed at top, bottom, left, or right of the screen.

Expression System

The taskbar has a built-in expression language for dynamic content: - Arithmetic: `+`, `-`, `*`, `/`, `%` - Comparisons: `>`, `>=`, `<`, `<=`, `=` - Conditionals: `If <expr> <action> [else <action>]` - Variables: `Var name = expression` - Functions: `Read()`, `Exec()`, `Time()`, `Disk()`, etc.

CSS Styling

Appearance is controlled via CSS (GTK+ compatible). CSS can be in a separate `.css` file or inline in the config. Widget types and styles are targeted using GTK named widget syntax:

```
label#clock {
    font-size: 12px;
    color: white;
}
```

Menu System

Custom menus with items, separators, and submenus:

```
menu "MyMenu" {
    item "Terminal" { action = Exec "lxterminal" }
    separator {}
    item "Shutdown" { action = Exec "wmpoweroff" }
}
```

Key Functions Available in Expressions

Function	Purpose
`Read(file)`	Read file contents
`Exec(cmd)`	Execute command, return output
`Time(format)`	Current time/date
`Disk(path,metric)`	Disk usage info
`TestFile(path)`	Check if file exists
`Str(expr)`	Convert to string
`Replace(str,old,new)`	String replacement
`SetValue(widget,val)`	Update widget content
`SetStyle(widget,css)`	Change widget CSS class
`SwayCmd(cmd)`	Send command to compositor
`MpdCmd(cmd)`	Control music player

Troubleshooting

Taskbar not appearing: Check that the taskbar is started in the EEDesktop autostart. Run `sfwbar` from terminal to see errors.

Taskbar wrong size/position: Edit sfwbar.config, check `edge` and `size` properties on the bar definition.

System tray icons missing: The tray widget requires apps that support the StatusNotifierItem protocol. Some older apps only support the X11 system tray and won't appear under Wayland.

EEFiles

Binary: `/usr/bin/eeos-files` (symlink to the file manager binary) **Config directory:** `~/.config/spacefm/` **System config:** `/etc/spacefm/spacefm.conf` **Desktop entries:** `file_app.desktop` (desktop icon) **Autostart:** EEFiles desktop manager mode (optional)

What It Is

EEFiles is a multi-panel tabbed file manager with a built-in virtual file system, udev-based device manager, customizable menu system, and deep bash integration. It can also manage the desktop (icons, wallpaper, right-click menu).

Panel System

EEFiles windows contain up to four independently-configured panels: - Panels 1-2: top half of the window - Panels 3-4: bottom half - Any panel can be shown or hidden independently - Each panel supports multiple folder tabs - Each panel has optional side panes: Devices, Bookmarks, Directory Tree

Panel memory maintains four distinct configuration states based on which panels are visible together, remembering column widths, side pane sizes, and toolbar visibility for each layout.

Path Bar Features

The path bar at the top of each panel supports: - **Tab completion** for directory and file names - **Breadcrumb navigation** (Ctrl+Click on path segments) - **Bash command execution** using prefixes: - `$command` – run as task with progress - `&command` – run in background - `+command` – run in terminal - `!command` – run as root - **Protocol URLs:** `ftp://`, `smb://`, `ssh://`, `nfs://` for network mounts - **Pattern selection:** `%*.jpg` selects all JPG files in current directory - **Find-As-You-Type:** Start typing to search in current directory (glob patterns work)

File Operations

- **Copy/Move/Link:** Via right-click menu, keyboard shortcuts (Ctrl+C/V/X), or drag-and-drop
- **Rename:** Single file rename dialog with path editing, or multi-file batch rename
- **New file/folder:** From templates directory (`~/Templates/`)
- **Permissions:** Change owner, group, permissions via Properties dialog
- **Root operations:** Right-click > Root operations for privileged file management

Bookmarks

- Organize folder shortcuts into menus and submenus
- Drag-and-drop reordering
- Bookmarks can execute socket commands (open specific panel, automate tasks)
- Import/export GTK bookmarks

- "Follow Dir" mode highlights the bookmark matching the current directory

Device Manager

Built-in udev-based device manager in the Devices side pane: - **Single-click mount/unmount** for USB drives, partitions, optical media - **Auto-mount** option for removable media - **Format** drives (right-click > Format – requires root) - **Filesystem check** (right-click > Check) - **Backup/restore** via FSArchiver or Partimage - **Mount options** customizable per filesystem type - **Custom handlers** for special devices or protocols

Desktop Manager

When started in desktop mode, EEFiles manages: - Desktop icons with transparent background support - Desktop right-click menu (fully customizable) - Desktop wallpaper - Works with the EEDesktop compositor and other minimal window managers

Custom Commands and Plugins

Design Mode (available from the menu) allows: - Renaming or hiding any menu/toolbar item - Adding custom commands that execute bash scripts - Bash variable substitution: `%d` (directory), `%f` (file), `%v` (device), `%l` (label) - Creating plugins (exportable custom commands and configurations) - Socket commands for external script control of EEFiles

Task Manager

File operations run as tracked tasks: - Progress bars for copy/move operations - Pause/resume/stop controls - Copy speed statistics - Queue management (operations on the same device wait) - Error handling with optional popups

Troubleshooting

File manager won't open: Try running the file manager from a terminal to see errors.

Desktop icons not showing: Ensure the EEFiles desktop manager autostart is enabled. Check that the `Hidden=` field is not set to `true` in the autostart desktop entry.

Can't mount USB drive: Click the device in the Devices pane. If it fails, check `dmesg` for errors. The device may need a filesystem (use right-click > Format). For NTFS, ensure `ntfs-3g` is available.

Right-click menu items missing: EEFiles menus are fully customizable. Items may have been hidden via Design Mode. Right-click the menu bar > Show Hidden to reveal them.

Part 13: Network Management

EENetwork

Daemon: `/usr/sbin/connmand` **CLI:** `/usr/bin/connmanctl` **GUI:** EENetwork GUI (`connman-gtk`) or frisbee/peasywifi **Config:** `/var/lib/connman/` (per-service profiles) **Init script:** `/etc/init.d/connman`

What It Is

EENetwork is the network management service in EEOS. Lightweight and modular, it handles Ethernet, WiFi, Bluetooth tethering, VPN, and mobile broadband through a plugin architecture.

connmanctl Command Reference

`connmanctl` is the CLI tool for EENetwork. Run without arguments for interactive shell, or pass commands directly.

BASIC COMMANDS

```
# show overall status
connmanctl state

# list available technologies (wifi, ethernet, bluetooth)
connmanctl technologies

# enable/disable a technology
connmanctl enable wifi
connmanctl disable wifi

# list available services (networks)
connmanctl services

# show details for a specific service
connmanctl services wifi_<id>

# scan for WiFi networks
connmanctl scan wifi
```

CONNECTING TO WIFI

Interactive shell method:

```
# connmanctl
connmanctl> enable wifi
connmanctl> agent on
connmanctl> scan wifi
connmanctl> services
  MyNetwork    wifi_abc123_managed_psk
connmanctl> connect wifi_abc123_managed_psk
  Agent RequestInput wifi_abc123_managed_psk
  Passphrase = [enter password]
connmanctl> exit
```

Once connected, EENetwork stores the credentials in `/var/lib/connman/wifi_<id>/settings` and auto-connects on subsequent boots.

STATIC IP CONFIGURATION

```
connmanctl config wifi_<id> --ipv4 manual 192.168.1.100 255.255.255.0 192.168.1.1
connmanctl config wifi_<id> --nameservers 8.8.8.8 8.8.4.4
```

TETHERING

```
connmanctl tether wifi on MyHotspot password123
connmanctl tether wifi off
```

Profile Storage

EENetwork stores connection profiles in `/var/lib/connman/<service_id>/settings`:

```
[wifi_abc123_managed_psk]
Name=MyNetwork
SSID=4d794e6574776f726b
Frequency=2437
Favorite=true
AutoConnect=true
Passphrase=MyPassword123
IPv4.method=dhcp
```

GUI Frontends

EEOS includes multiple graphical frontends for EENetwork:

- **EENetwork GUI:** Full GTK3 interface for managing all connection types
- **frisbee:** Lightweight WiFi-focused connection tool
- **peasywifi:** Simple WiFi tray applet

These are accessed via the Connection Wizard (`connectwizard`), which auto-selects the appropriate frontend based on what's installed and the connection type needed.

Troubleshooting

No network after boot: 1. Check `connmanctl technologies` – is ethernet/wifi enabled? 2. Check physical cable/WiFi adapter 3. Try `connmanctl scan wifi` then `connmanctl services` 4. Check `dmesg | grep -i net` for driver issues

WiFi connects but no internet: 1. `connmanctl services` – check State is "online" not just "ready" 2. Check DNS: `nslookup google.com` 3. Try static DNS: `connmanctl config <id> --nameservers 8.8.8.8`

Saved WiFi won't auto-connect: Check `/var/lib/connman/<service>/settings` – `AutoConnect=true` and `Favorite=true` must both be set.

EENetwork conflicts with other network managers: EEOS uses EENetwork exclusively. If other network managers somehow got installed, they'll conflict. Disable them: `systemctl disable NetworkManager`.

Part 14: Audio System

EEAudio

Daemon: `/usr/bin/pipewire` **PulseAudio compat:** `/usr/bin/pipewire-pulse` **Session manager:** `/usr/bin/wireplumber` **CLI control:** `wpctl`, `pw-cli`, `pactl`

What It Is

EEAudio is a unified audio server that replaces older audio systems with a single, modern stack. It provides full compatibility with PulseAudio applications plus low-latency audio when needed.

Key Commands

```
# list all audio devices
wpctl status

# set output volume
wpctl set-volume @DEFAULT_AUDIO_SINK@ 0.5      # 50%
wpctl set-volume @DEFAULT_AUDIO_SINK@ 5%+      # increase 5%
wpctl set-volume @DEFAULT_AUDIO_SINK@ 5%-      # decrease 5%

# mute/unmute
wpctl set-mute @DEFAULT_AUDIO_SINK@ toggle

# list sinks (outputs) and sources (inputs)
pactl list sinks short
pactl list sources short

# set default output
wpctl set-default <id>
```

Configuration

Configuration is in `/etc/pipewire/` and `~/.config/pipewire/`. ALSA backend configuration in `/etc/alsa/conf.d/`.

Troubleshooting

No sound: 1. `wpctl status` – check for active sinks 2. `alsamixer` – check nothing is muted (MM means muted, press M to unmute) 3. Check that EEAudio is running: `pgrep pipewire` 4. If using HDMI audio, it may be a separate sink – use `wpctl set-default` to switch

Sound crackling or stuttering: Usually a buffer underrun. Check CPU load. If running in Copy-to-RAM mode, this shouldn't happen. On direct USB boot, it may indicate the USB is too slow for simultaneous audio and disk access.

Appendix A: Desktop Entry Names

The official `.desktop` `Name=` field for each EE application, as configured in the SFS:

Desktop file	Display Name
`eeai-diag.desktop`	EEDiagnostics
`eeos-sysinfo.desktop`	EESysInfo
`eeos-kview.desktop`	EEKView
`eeos-control.desktop`	EEControl
`eeos-setup.desktop`	EESetup
`eeos-setup-hub.desktop`	EESetup Hub
`eeos-display-wizard.desktop`	EEDisplay
`eeos-default-apps.desktop`	EEDefaults
`eeos-pkg.desktop`	EEPackages
`eeos-package-manager.desktop`	EEPackages
`eeos-save-manager.desktop`	EESession
`eeos-session-backup.desktop`	EEBackup
`eeos-swap.desktop`	EESwap
`eeos-installer.desktop`	EEInstaller
`eeos-universal-installer.desktop`	EEInstaller
`eeos-remaster-classic.desktop`	EERemaster
`eeos-stick-installer.desktop`	EEStick Installer
`eeos-e4-stick-installer.desktop`	EE4Stick Installer
`eeos-e3-stick-installer.desktop`	EE3Stick Installer
`eeos-f2-stick-installer.desktop`	EEF2Stick Installer
`eeos-archive.desktop`	EEArchive

Desktop file	Display Name
`eeos-advert-blocker.desktop`	EEAdBlocker
`eeos-md5sum.desktop`	EEChecksum
`eeos-pdf-convert.desktop`	EEPDF
`eeos-help.desktop`	EEHelp
`eeos-network-shares.desktop`	EENetwork Shares
`eeos-event-manager.desktop`	EEEvents
`eeos-utility-suite.desktop`	EEUtilities
`eeos-disk-installer.desktop`	EEDisk Installer
`eeos-installers.desktop`	EEInstallers

Appendix B: File Paths Quick Reference

What	Where
EE app binaries	`/usr/local/bin/eeos-*`, `/usr/local/bin/eeai-*`
EE sbin tools	`/usr/sbin/eeos-*`
EESysInfo app dir	`/usr/local/eeos-sysinfo/`
EEKView app dir	`/usr/local/eeos-kview/`
EEControl app dir	`/usr/local/eeos-control/`
EEPackages app dir	`/usr/local/eeos-pkg/`
EESetup Hub app dir	`/usr/local/eeos-setup-hub/`
Desktop entries	`/usr/share/applications/eeos-*.desktop`
Autostart entries	`/etc/xdg/autostart/`
Kiosk config	`/etc/eeos/kiosk.conf`
Remote agent config	`/etc/eeos/remote-agent.conf`
Module config	`/etc/rc.d/MODULESCONFIG`
Boot state	`/etc/rc.d/EEOSSTATE`
Boot constraints	`/etc/rc.d/BOOTCONFIG`, `/etc/rc.d/BOOTCONSTRAINED`

What	Where
Save state	`eeossave.4fs` or `eeossave/` on boot USB
HHFE	`/opt/hhfe/`
DOSBox config	`/opt/hhfe/.dosbox/dosbox-hhfe.conf`
HHFE invocation	`/opt/hhfe/INVOKE.TXT`
Firmware	`/lib/firmware/`
SFS modules	`/mnt/live/boot/` or boot USB root
EEOS reference docs	`/usr/share/eeos/`
Xorg config	`/etc/X11/xorg.conf`, `/etc/X11/xorg.conf.d/`
EEDesktop config	`~/.config/labwc/`
EETaskbar config	`~/.config/sfwbar/`
conky config	`~/.conkyrc`
Remote log	`/var/log/eeos-remote.log`
HHFE log	`/var/log/eeos-hhfe.log`
Diagnostic reports	`/tmp/eeos-diagnostic-report-*.txt`

Appendix C: General Troubleshooting Matrix

Symptom	First check	Second check	Third check
Machine won't boot	Different USB port	BIOS boot order	Maximum Compatibility GRUB entry
Black screen after boot	Wait 30 seconds	Safe Video GRUB entry	NVIDIA Safe GRUB entry
Desktop is sluggish	RAM usage (EESysInfo)	Swap status (EESwap)	Copy to RAM boot mode
HHFE won't start	EEDiagnostics status	Repair menu: restart HHFE	Check INVOKE.TXT
Network not working	Cable/WiFi physical	EEDiagnostics network	Repair menu: restart network
Changes not saving	Check boot mode (not RAM-only)	USB free space	EESession: merge or resize
Display resolution wrong	EEDisplay	Safe Video boot	Force display setup boot
Sound not working	EESysInfo: Audio devices	ALSA mixer (alsamixer)	

Symptom	First check	Second check	Third check
			Check EEAudio: `pw-cli info`
USB device not detected	EEEvents log	`dmesg | tail -20`	EEKView: check driver
WiFi not connecting	Connection wizard	`iwlist scan`	Check firmware in EEKView
Printer not working	CUPS web UI (localhost:631)	EEControl: Print Setup	Check USB connection
Package install fails	`apt-get -f install`	Check network	Check sources.list

Appendix D: Package Management Internal Architecture

The EEOS package system uses a standardized database format that handles EE (.pet), DEB, RPM, and SFS packages uniformly.

Database Format

Package databases are stored in `/root/.packages/` with pipe-delimited fields:

```
pkgname|version|build|category|size|path|filename|dependencies|description|repo
```

Example:

```
firefox|115.0|1|Internet|85000K|ee_packages-trixie|firefox-115.0.pet|
+libgtk-3-0,+libnss3|Web browser|trixie
```

Database Files

File	Contents
`eeos-installed-packages`	Packages built into the SFS image
`user-installed-packages`	Packages the user installed
`Packages-*`	Repository package lists
`DISTRO_PKGS_SPECS`	Compatible distro information
`DISTRO_EE_REPOS`	EE package repository URLs
`*.files`	File lists for each installed package

Package Types

Type	Extension	Description
EE (.pet)	`.pet`	EEOS native format. May be split into `_DEV`, `_DOC`, `_NLS` sub-packages
DEB	`.deb`	Debian packages from trixie repositories
SFS	`.sfs`	Squashfs modules overlaid onto root (not extracted)
RPM	`.rpm`	Can be converted and installed

Dependency Resolution

Dependencies are listed with `+` prefix in the database. The package manager resolves them recursively, presenting a hierarchy view showing what depends on what. Cross-format resolution works because all packages use the same standardized database format.

Appendix E: SFS Module Layer System

EEOS uses a layered filesystem architecture where multiple read-only squashfs images (SFS modules) are stacked together with a writable overlay on top.

How Layers Work

```
        +--------------------------+
        |  RAM overlay (tmpfs)     |  <- writable, all changes go here
        +--------------------------+
        |  eeossave (persistent)   |  <- optional, saved to USB
        +--------------------------+
        |  eeos_system.sfs         |  <- main OS (495MB)
        +--------------------------+
        |  eeos_modules.sfs        |  <- kernel modules
        +--------------------------+
        |  eeos_firmware.sfs       |  <- hardware firmware
        +--------------------------+
        |  eeos_drivers.sfs        |  <- driver packages
        +--------------------------+
        |  eeos_base.sfs           |  <- minimal base system
        +--------------------------+
        |  eeos_kbuild.sfs         |  <- kernel build headers
        +--------------------------+
```

Applications see a single unified root filesystem. Files from higher layers override files in lower layers. The overlay captures all changes (new files, modifications, deletions).

Whiteout Files

When a file from a read-only layer is "deleted," the overlay creates a whiteout file (`.wh.<filename>`) that hides the original. The file still exists in the SFS but is invisible to the running system.

Loading Additional SFS Modules

SFS modules can be loaded at runtime using `sfs_load` : 1. The SFS is mounted as a loop device 2. The mount is added to the overlay's lower directory stack 3. Contents become immediately available without extraction or installation 4. Unloading reverses the process

This is how optional components (language packs, development tools, additional driver sets) are added without modifying the base system.

Persistence Modes

Mode	How changes persist	When to use
Savefile (`.4fs`)	Fixed-size ext4 file on USB	FAT32/NTFS USB sticks
Savefolder	Directory on USB	ext4/f2fs USB sticks
RAM-only	No persistence	Testing, kiosk, ephemeral
Copy-to-RAM	USB can be removed after boot	Centers with 8GB+ RAM

EEOS Save Modes

The init system uses save modes to determine persistence behavior:

Mode	Behavior
5	RAM-based save, auto-writes on change. Used before a save file is created.
12	Write-on-demand to save. Used for internal drive installs.
13	Write on schedule or shutdown only. Used for USB to limit write cycles.

EEOS 1.0 – Built for EESystem centers. Everything works offline. Nothing phones home.