v.0.0.1

# ItadOS

Function Reference

Karl-Markus Oismann
7-12-2025

# Contents

# config.sh

**Local dependencies:** None
**Description:**
Defines environment variables that control user interaction, erasure behaviour, reporting style, and other itadOS settings.

## Configuration Variables

**MANUAL_USER_CONF**

- **Description:** Toggles whether the user is prompted for manual settings on boot.
- **Options:** on, off
- **Default:** on

---

**ASSET_CONF**

- **Description:** Sets the method of asset tag acquisition. Serial setting inserts asset tag automatically as serial number of the computer.
- **Options:** asset, serial
- **Default:** asset

---

**ERASURE_NAME_CONF**

- **Description:** Custom name to use for the erasure method (shown in reports).
- **Example:** "itadOS"

---

**ITADOS_VERSION_CONF**

- **Description:** itadOS version shown in the report.
- **Example:** "itadOS v.0.0.1"

---

**ERASURE_LOGO_CONF**

- **Description:** Path to the logo used in the erasure report.

### System Info Output

**SYSTEM_SPEC_CONF**

- **Description:** Determines how system specifications are displayed on the report.
- **Options:** full (uses lshw -short), min (short readable output)
- **Default:** min

### Service Information

All fields can be set by the user or left blank for manual entry:

- **TECHNICIAN_CONF**
- **PROVIDER_CONF**
- **LOCATION_CONF**
- **CUSTOMER_CONF**
- **JOBNR_CONF**

### Erasure Specification

**ERASURE_SPEC_CONF**

- **Description:** Specifies the default erasure method. Auto attempts purge and falls back to clear if needed.
- **Options:** purge, clear, auto, skip
  - **Purge:** Most secure erasure methods; Not supported by all disks
  - **Clear:** Less secure erasure methods; Supported by majority of disks
  - **Auto:** If purge fails, falls back to clear.
  - **Skip:** Skips erasure.
- **Default:** auto

---

**VERIFICATION_CONF**

- **Description:** Controls how verification is done after erasure. Partial reads first and last 10% of disk.
- **Options:** full, partial, skip
  - **Full:** Scans entire disks for zero pattern.
  - **Partial:** Scans first and last 10% of disk for zero pattern.
  - **Skip:** Skips verification.
- **Default:** full

## Other Options

**CHECK_ZERO_PATTERN_AND_OVERWRITE_CONF**

- **Description:** If enabled, fills disk with zeros if zero-pattern check fails. This is only checked after erasure is completed using methods such as secure erase, crypto erase etc.
- **Options:** on, off
- **Default:** on

---

**FILTER_BOOT_DISK_CONF**

- **Description:** If enabled, prevents the boot disk from being shown as erasable.
- **Options:** on, off
- **Default:** on

---

**SMART_TEST_CONF**

- **Description:** Defines SMART test mode for drive health.
- **Options:** short, long, skip
  - **Short:** quick check, takes up to 2 minutes.
  - **Long:** More comprehensive test can take hours.
  - **Skip:** skips health check.
- **Default:** short

---

**SUSPEND_CONF**

- **Description:** Suspends the computer before erasure attempts.
- **Options:** on, off
- **Default:** off

---

**AUTO_ERASURE_CONF**

- **Description:** Automatically erases all attached disks (except boot disk if filter is on).
- **Options:** on, off
- **Default:** off

# lib/initialize.sh

**Local dependencies:** None
**Description:**
Prepares the runtime environment for itadOS by identifying the boot disk, clearing temporary files, and setting environment variables. Includes helper functions to identify disk properties.

---

## isDiskRemovable

- **Description:** Checks whether the specified disk is marked as removable.
- **Parameters:**
  - $1 = Disk name (e.g., sdb)
- **Returns:**
  - 0 if the disk is removable
  - 1 if the disk is not removable
- **Behavior:**
  - Reads /sys/class/block/<disk>/removable to determine if the device is removable.

---

## amountOfPartitions

- **Description:** Counts the number of partitions on a disk.
- **Parameters:**
  - $1 = Disk name (e.g., sdb)
- **Returns:**
  - Number of partitions
- **Behavior:**
  - Uses ls and awk on /sys/class/block/<disk> to count partition entries.

---

## partitionLabel

- **Description:** Retrieves the label of the first partition on a given disk.
- **Parameters:**
  - $1 = Disk name (e.g., sdb)
- **Returns:**
  - Partition label string (e.g., ITADOS)
- **Behavior:**
  - Uses lsblk and awk to match disk label based on its parent.

---

### findItadOS

- **Description:** Attempts to identify the boot disk by scanning mounted filesystems.
- **Parameters:**
  - None
- **Returns:**
  - Nothing
- **Behavior:**
  - Currently relies on lsblk output; deep-search logic is commented out.

---

### getBootDisk

- **Description:** Identifies and exports the boot disk and its serial number to a temporary file.
- **Parameters:**
  - None
- **Returns:**
  - Exports BOOT_DISK and BOOT_SERIAL
- **Behavior:**
  - Loops through detected disks, checks if removable and labeled ITADOS, stores results in lib/files/tmp/bootDisk.txt.

---

### initialize

- **Description:** Sets up necessary folders, clears previous temp files, and initializes global state variables.
- **Parameters:**
  - None
- **Returns:**
  - Nothing
- **Behavior:**
  - Creates lib/files/tmp/logs/ if missing
  - Clears various temp directories and files
  - Calls getBootDisk
  - Initializes global variables like ERRORS, HIDDEN_TRIGGERED, CONFIRMED, etc.

# lib/userConfig.sh

**Local dependencies:** config.sh (for initial values)
**Description:**
Provides a user-friendly menu for setting and overriding erasure settings and service-related configuration using interactive prompts. Values are stored in exported environment variables and override values from config.sh.

---

## technician

- **Description:** Prompts the user to enter the technician's name. Overrides TECHNICIAN_CONF.
- **Parameters:**
  - None
- **Returns:**
  - Updated environment variable TECHNICIAN_CONF
- **Behavior:**
  - Uses whiptail --inputbox; handles cancel and empty input validation.

---

## customer

- **Description:** Prompts the user to enter the customer's name. Overrides CUSTOMER_CONF.
- **Parameters:**
  - None
- **Returns:**
  - Updated CUSTOMER_CONF
- **Behavior:**
  - Uses whiptail --inputbox; handles cancel and empty input validation.

---

## provider

- **Description:** Prompts the user to enter the provider's name. Overrides PROVIDER_CONF.
- **Parameters:**
  - None
- **Returns:**
  - Updated PROVIDER_CONF
- **Behavior:**
  - Uses whiptail --inputbox; handles cancel and empty input validation.

---

### location

- **Description:** Prompts the user to enter the location. Overrides LOCATION_CONF.
- **Parameters:**
  - None
- **Returns:**
  - Updated LOCATION_CONF
- **Behavior:**
  - Uses whiptail --inputbox; handles cancel and empty input validation.

---

### job

- **Description:** Prompts the user to enter the job number. Overrides JOBNR_CONF.
- **Parameters:**
  - None
- **Returns:**
  - Updated JOBNR_CONF
- **Behavior:**
  - Uses whiptail --inputbox; handles cancel and empty input validation.

---

### erasure_spec

- **Description:** Lets the user choose an erasure specification (clear, purge, auto, skip).
  - **clear**: Less secure erasure methods; Supported by majority of disks.
  - **purge**: Most secure erasure methods; Not supported by all disks.
  - **auto**: If purge fails, falls back to clear.
  - **skip**: Skips erasure.
- **Parameters:**
  - None
- **Returns:**
  - Updated ERASURE_SPEC_CONF
- **Behavior:**
  - Handles user input via whiptail.

---

### verification_spec

- **Description:** Lets the user choose the verification method (full, partial, skip).
  - **full:** Scans entire disks for zero pattern.
  - **partial:** Scans first and last 10% of disk for zero pattern.
  - **skip:** Skips verification.
- **Parameters:**
  - None
- **Returns:**
  - Updated VERIFICATION_CONF
- **Behavior:**
  - Handles user input via whiptail.

### asset_tag_setting

- **Description:** Allows user to set asset tag mode (asset, serial).
  - **asset:** Manual asset tag entry.
  - **serial:** Asset tag automatically set as serial number of host computer.
- **Parameters:**
  - None
- **Returns:**
  - Updated ASSET_CONF
- **Behavior:**
  - Handles user input via whiptail.

---

### suspend_setting

- **Description:** Sets suspension behavior before erasure (on, off).
- **Parameters:**
  - None
- **Returns:**
  - Updated SUSPEND_CONF
- **Behavior:**
  - Handles user input via whiptail.

---

### smart_setting

- **Description:** Sets SMART test type (short, long, skip).
  - **short:**
  - **long:**
  - **skip:**
- **Parameters:**
  - None
- **Returns:**
  - Updated SMART_TEST_CONF
- **Behavior:**
  - Handles user input via whiptail.

---

### boot_filter_setting

- **Description:** Toggles hiding boot disk from erasure list.
- **Parameters:**
  - None
- **Returns:**
  - Updated FILTER_BOOT_DISK_CONF
- **Behavior:**
  - Handles user input via whiptail.

### auto_overwrite_setting

- **Description:** Sets auto-overwrite behavior.
- **Parameters:**
  - None
- **Returns:**
  - Updated CHECK_ZERO_PATTERN_AND_OVERWRITE_CONF
- **Behavior:**
  - Handles user input via whiptail.

---

### auto_erasure_setting

- **Description:** Sets auto-erasure on boot (on, off).
- **Parameters:**
  - None
- **Returns:**
  - Updated AUTO_ERASURE_CONF
- **Behavior:**
  - Handles user input via whiptail.

---

### getServiceInfoChoice

- **Description:** Multi-select menu for modifying service info fields.
- **Parameters:**
  - None
- **Returns:**
  - 0 if selections were made,
  - 1 if canceled,
  - 2 if OK pressed without selection
- **Behavior:**
  - Uses whiptail checklist to gather choices.

---

### getErasureSpecChoice

- **Description:** Menu to select erasure and verification settings.
- **Parameters:**
  - None
- **Returns:**
  - 0 if selections were made,
  - 1 if canceled,
  - 2 if OK pressed without selection
- **Behavior:**
  - Uses whiptail checklist to gather choices.

---

### getOtherChoice

- **Description:** Menu for selecting miscellaneous settings (e.g., auto overwrite, health check).
- **Parameters:**
  - None
- **Returns:**
  - 0 if selections were made,
  - 1 if canceled,
  - 2 if OK pressed without selection
- **Behavior:**
  - Uses whiptail checklist to gather choices.

---

### confirmation

- **Description:** Summarizes all current settings and asks the user to confirm.
- **Parameters:**
  - None
- **Returns:**
  - If confirmed, sets CONFIRMED=yes
- **Behavior:**
  - Displays formatted message using whiptail --yesno.

---

### serviceInfoConfig

- **Description:** Main loop for processing service info menu selection and invoking relevant functions.
- **Parameters:**
  - None
- **Returns:**
  - None
- **Behavior:**
  - Uses whiptail checklist to gather choices.

---

### erasureSpecChoiceConfig

- **Description:** Handles logic for erasure and verification configuration.
- **Parameters:**
  - None
- **Returns:**
  - None
- **Behavior:**
  - Calls respective functions based on checklist selection.

---

### otherChoiceConfig

- **Description:** Handles advanced and miscellaneous configuration options.
- **Parameters:**
  - None
- **Returns:**
  - None
- **Behavior:**
  - Dispatches to correct settings based on selected items.

---

### menu

- **Description:** Root menu loop for launching submenus and managing the flow until the configuration is confirmed.
- **Parameters:**
  - None
- **Returns:**
  - None
- **Behavior:**
  - Displays core categories (Service Info, Erasure Spec, Other, Continue) and invokes submenus accordingly.

## lib/interaction.sh

**Local dependencies:** none
**Description:**
Handles user interaction during and after the erasure process, including cancellation, shutdown, and retry options.

---

### removeBootInfo

- **Description:** Deletes temporary file containing boot disk information.
- **Parameters:**
  - None
- **Returns:**
  - None
- **Behavior:**
  - Removes lib/files/tmp/bootDisk.txt.

---

### shutdownNow

- **Description:** Shuts down the system immediately.
- **Parameters:**
  - None
- **Returns:**
  - None
- **Behavior:**
  - Sets RUNNING="false"
  - Calls shutdown now to power off the device

---

### reasonForCancel

- **Description:** Prompts the user for a reason when canceling erasure.
- **Parameters:**
  - None
- **Returns:**
  - Sets the global variable REASON_FOR_CANCEL
- **Behavior:**
  - Displays an input box using whiptail
  - If user provides no input and presses OK, re-prompts
  - If user cancels, sets reason as "Not given."
  - If input is valid, exports it as REASON_FOR_CANCEL

---

### finalMenu

- **Description:** Displays a menu after erasure is completed, offering post-operation choices.
- **Parameters:**
  - None
- **Returns:**
  - None
- **Options:**
  - **Shutdown:** Powers off the computer immediately using shutdownNow
  - **Retry:** Restarts erasure process.
  - **Exit:** Returns user to CLI and removes boot info
- **Behavior:**
  - Uses whiptail --menu to offer user 3 post-erasure options
  - Handles cleanup or shutdown depending on selection

## lib/reportToUSB.sh

**Local dependencies:** /lib/files/tmp/usbDrives.txt
**Description:**
Handles detection, mounting, optional formatting, and report transfer to a user-selected USB drive after erasure. Ensures reports are saved externally in a user-friendly and error-resilient way.

---

### saveToUSB

- **Description:** Moves erasure reports to the selected USB drive mount point.
- **Parameters:**
  - $1 = USB drive name (e.g., sdb)
- **Returns:**
  - None
- **Behavior:**
  - Verifies reports exist in lib/files/reports/
  - Moves all reports to the selected USB mount directory
  - Prompts user if report transfer fails
  - If successful, displays confirmation and unmounts the USB drive

---

### reportToUSB

- **Description:** Prompts the user to choose a USB device, mounts it, optionally formats it, and calls saveToUSB.
- **Parameters:**
  - None
- **Returns:**
  - None
- **Behavior:**
  - Refreshes and lists USB drives using lsblk
  - Displays available USB drives using a whiptail radiolist
  - Mounts selected USB to /media/<device>
  - If mounting fails and disk is unformatted, offers to format it
  - Formats using sfdisk and mkfs.exfat
  - If formatting or mounting eventually succeeds, continues to saveToUSB
  - Offers to delete stale mount directories if present
- **Options:** (shown to user during execution via whiptail)
  - List of USB drives (e.g., sdb, sdc) with detected model names
  - If unformatted:
    - **Yes**: Format to exFAT
    - **No**: Abort formatting
  - If mountpoint already exists:
    - **Yes**: Delete stale mountpoint and retry
    - **No**: Abort transfer

## lib/getAssetTag.sh

**Local dependencies:** reportToUSB.sh
**Description:**
Handles user input for assigning a system asset tag. Prevents duplicate tags and allows handling of existing reports (delete, transfer, or re-enter tag). Ensures every erasure session has a unique identifier.

---

### ifReportExists

- **Description:** Checks if a report for the entered asset tag already exists. If so, prompts the user to choose what to do with it.
- **Parameters:**
  - None
- **Returns:**
  - May modify or delete report files under lib/files/reports/
  - May call getAssetTag or reportToUSB
- **Behavior:**
  - If a PDF report named <ASSET_TAG>.pdf exists, shows a whiptail menu
  - Responds to user actions and repeats until a valid choice is made
- **Options:**
  - **Delete report:** Removes both .pdf and .xml reports for the asset tag
  - **Transfer report:** Launches USB transfer via reportToUSB
  - **New asset tag:** Prompts for a new asset tag via getAssetTag; if it still exists, asks again

---

### getAssetTag

- **Description:** Prompts the user to enter an asset tag and stores it as a global variable.
- **Parameters:**
  - None
- **Returns:**
  - Sets global ASSET_TAG
- **Behavior:**
  - Displays whiptail input box with instruction to remove itadOS USB
  - Exports user input as ASSET_TAG regardless of validation

---

### getAssetTagRQ

- **Description:** Prompts the user for an asset tag and validates the input is not empty. Handles existing report conflicts.
- **Parameters:**
  - None
- **Returns:**
  - Sets global ASSET_TAG
- **Behavior:**
  - Calls getAssetTag to prompt for input
  - Loops until a non-empty tag is entered
  - Calls ifReportExists to check for existing reports and handle user's choice accordingly

## lib/log.sh

**Local dependencies:** Requires that the LOG_FILE environment variable is set and writable.
**Description:**
Appends timestamped log messages to a predefined log file to support auditing, debugging, and system monitoring.

---

### log

- **Description:** Writes a timestamped message to the log file specified by LOG_FILE.
- **Parameters:**
  - $1 = Message to be logged
- **Returns:**
  - None
- **Behavior:**
  - Captures the current timestamp in the format YYYY-MM-DD HH:MM:SS
  - Appends the timestamp and message to the file at $LOG_FILE
  - Assumes $LOG_FILE has already been exported by another script or config

# lib/getAttachedDisks.sh

**Local dependencies:** log.sh

**Description:** Detects and exports attached disk information to temporary files. Optionally filters out the boot disk if FILTER_BOOT_DISK_CONF is set to 'on'. The total number of disks is stored in an exported variable.

---

## getAttachedDisks

- **Description:** Retrieves and logs all connected block storage disks and writes their details to a temp file. Can optionally filter out the boot disk to prevent accidental erasure.
- **Parameters:**
  - None
- **Returns:**
  - ATTACHED_DISKS (file path as exported variable)
  - ATTACHED_DISKS_COUNT (exported disk count)

    **Behavior:**
  - If FILTER_BOOT_DISK_CONF is 'on', calls filterBootUSB() to remove boot disk from list
  - Otherwise, collects all disks of type disk using lsblk
  - Saves disk info to lib/files/tmp/attachedDisks.txt
  - Logs attached disk data and updates global environment variables accordingly
- **Options:** (based on FILTER_BOOT_DISK_CONF)
  - **on**: Filters out the boot disk by serial and device name
  - **off**: Lists all connected disks, including boot media

---

## filterBootUSB

- **Description:** Filters out the boot device from the disk list using its name and serial number.
- **Parameters:**
  - None
- **Returns:**
  - Updates ATTACHED_DISKS with non-boot disks.
  - Sets BOOT_DISK_WARNING if boot disk is recorded on spec sheet.
- **Behavior:**
  - Compares each device's name and serial against BOOT_DISK and BOOT_SERIAL
  - Only appends non-boot disks to the output file
  - If at least one disk was excluded, sets BOOT_DISK_WARNING with the name of the filtered boot disk
  - Exports BOOT_DISK_WARNING to notify the UI or logs

## lib/getSystemInfo.sh

**Local dependencies:** none
**Description:** Collects detailed hardware information about the current system, including serial number, vendor, model, CPU, RAM, GPU(s), battery health, and disk information. Results are stored in global environment variables and an XML-formatted temporary file for minimal disk metadata.

---

### getSerialNumber

- **Description:** Retrieves and exports the system's serial number.
- **Parameters:**
  - None
- **Returns:**
  - Exports SERIAL_NUMBER
- **Behavior:**
  - Attempts dmidecode first, then /proc/cpuinfo
  - If both fail, defaults to "UNKNOWN"

---

### getModelVendor

- **Description:** Retrieves and exports the system manufacturer/vendor name.
- **Parameters:**
  - None
- **Returns:**
  - Exports SYSTEM_MANUFACTURER
- **Behavior:**
  - Uses dmidecode and /sys/class/dmi/id/sys_vendor
  - Falls back to "UNKNOWN" if not found

---

### getModelName

- **Description:** Retrieves and exports the system model name.
- **Parameters:**
  - None
- **Returns:**
  - Exports MODEL_NAME
- **Behavior:**
  - Uses dmidecode and /proc/cpuinfo as fallback
  - Defaults to "UNKNOWN" if both sources fail

---

### getCPUModel

- **Description:** Retrieves and exports the model name of the CPU.
- **Parameters:**
  - None
- **Returns:**
  - Exports CPU_MODEL
- **Behavior:**
  - Extracts first CPU model from /proc/cpuinfo
  - Falls back to "UNKNOWN" if not found

---

### getRAMSize

- **Description:** Retrieves and exports total system memory.
- **Parameters:**
  - None
- **Returns:**
  - Exports RAM_SIZE
- **Behavior:**
  - Uses free --giga -h to fetch memory in human-readable gigabyte format
  - Defaults to "UNKNOWN" if parsing fails

---

### getGPUModel

- **Description:** Retrieves and exports the name of the integrated GPU.
- **Parameters:**
  - None
- **Returns:**
  - Exports GPU_MODEL
- **Behavior:**
  - Uses lspci to search for VGA entries
  - Defaults to "UNKNOWN" if no match is found

---

### getDGPUModel

- **Description:** Retrieves and exports the name of the discrete GPU (if present).
- **Parameters:**
  - None
- **Returns:**
  - Exports DGPU_MODEL
- **Behavior:**
  - Uses lspci to find 3D controller
  - Defaults to "N/A" if not found

---

### getBatteryHealth

- **Description:** Determines battery presence and health percentage and exports the result.
- **Parameters:**
  - None
- **Returns:**
  - Exports BATTERY_HEALTH
- **Behavior:**
  - Checks /sys/class/power_supply/BAT0/ and alternative paths
  - If present, calculates health as (charge_full / charge_full_design) * 100%
  - If battery missing or invalid, exports "Missing battery", "UNKNOWN", or "N/A"

---

### getDiskInfo

- **Description:** Gathers minimal information for all attached disks and exports it to an XML-formatted file.
- **Parameters:**
  - None
- **Returns:**
  - Exports MIN_DISKS (path to XML-like file with disk metadata)
- **Behavior:**
  - Reads either attachedDisks.txt or attachedDisksFilter.txt based on FILTER_BOOT_DISK_CONF
  - If no disks are found, writes a warning into the file
  - Otherwise, extracts and formats name, size, type, and model for each disk
  - Appends results under <minDisk> tags for each disk

---

### GetSystemInfo

- **Description:** Master function that calls all individual system info collectors.
- **Parameters:**
  - None
- **Returns:**
  - None
- **Behavior:**
  - Executes the following in sequence:
    - getSerialNumber, getModelName, getModelVendor, getCPUModel, getRAMSize, getGPUModel, getDGPUModel, getBatteryHealth

## lib/getChosenDisks.sh

**Local dependencies:**

- placeSpecsToFolders, createDiskDir, TaskManager, reasonForCancel
- Temporary files: attachedDisks.txt, chosenDisks.txt, chosenDisksDesc.txt

**Description:**
Provides a whiptail UI for users to choose which attached disks to erase. Saves user selections to temporary files, creates directories for selected disks, and prepares specification data for each.

---

### diskDescription

- **Description:** Formats selected disk attributes for display in the selection UI.
- **Parameters:**
  - None (uses variables like size, type, and model)
- **Returns:**
  - A short string showing disk description
- **Behavior:**
  - Outputs a string like: 512G SATA SSD Samsung EVO

---

### getChosenDisks

- **Description:** Displays a checklist of available disks for the user to select for erasure.
- **Parameters:**
  - None
- **Returns:**
  - Exports CHOSEN_DISK_STATUS, CHOSEN_DISKS_COUNT, CHOSEN_DISKS, CHOSEN_DISKS_DESC
- **Behavior:**
  - Parses attachedDisks.txt to collect disk info
  - Dynamically determines disk type (HDD, SSD, eMMC, etc.)
  - Uses whiptail to display a multi-selection list
  - Saves chosen disks to chosenDisks.txt
  - Creates folders and populates them with disk specifications
  - Warns if not all disks are selected
- **Options:** (shown to user via checklist UI)
  - Device name + disk description per disk (e.g., sda: 512G SATA SSD Samsung)

---

### checkStatus

- **Description:** Evaluates the user's action after the disk selection prompt and continues or cancels erasure accordingly.
- **Parameters:**
  - None
- **Returns:**
  - None
- **Behavior:**
  - Based on CHOSEN_DISK_STATUS, it either:
    - Proceeds with verification and erasure
    - Asks user to select disks again if none chosen
    - Displays cancellation or error messages

---

### autoSelectAttachedDisks

- **Description:** Automatically selects all attached disks for erasure without prompting the user.
- **Parameters:** None
- **Returns:**
  - Populates chosenDisks.txt
  - Exports CHOSEN_DISKS_COUNT
- **Behavior:**
  - Reads from attachedDisks.txt
  - Adds all listed disks to chosenDisks.txt
  - Creates directories and populates specs
  - Initiates disk erasure process directly

# lib/getDiskInfo.sh

**Local dependencies:** log.sh

- Temporary directories: lib/files/tmp/chosenDisks/, lib/files/tmp/chosenDisks.txt

**Description:**
Provides functions to extract and structure disk metadata. Handles raw sector-based disk sizing, device specifications, and directory initialization for reporting and erasure processes.

---

## getDiskSectors

- **Description:** Returns the total number of sectors on a disk.
- **Parameters:**
  - $1 = Disk name (e.g., sda)
- **Returns:**
  - Sector count (e.g., 625142448)
- **Behavior:**
  - Reads from /sys/block/<disk>/size
  - Outputs numeric sector count

---

## getDiskSectorSize

- **Description:** Returns the size of each sector in bytes.
- **Parameters:**
  - $1 = Disk name
- **Returns:**
  - Sector size (e.g., 512)
- **Behavior:**
  - Reads from /sys/block/<disk>/queue/hw_sector_size

---

## getDiskSizeInBytes

- **Description:** Calculates total disk size in bytes.
- **Parameters:**
  - $1 = Disk name
- **Returns:**
  - Disk size in bytes
- **Behavior:**
  - Multiplies sector count by sector size using getDiskSectors and getDiskSectorSize

---

### getFullDiskSpecs

- **Description:** Saves complete disk specifications into a file.
- **Parameters:**
  - $1 = Disk name
- **Returns:**
  - Writes specifications.txt for the disk
- **Behavior:**
  - Uses lsblk -d with fields: KNAME,SIZE,SERIAL,TYPE,ROTA,TRAN,MODEL
  - Filters by disk name using grep

---

### getDiskType

- **Description:** Determines and saves the disk's type based on transport and rotation status.
- **Parameters:**
  - $1 = Disk name
- **Returns:**
  - Echoes and writes type to type.txt
  - Sets exported variable TYPE **(Why?)**
- **Behavior:**
  - Parses specifications.txt
  - Matches rules:
    - sata + ROTA=1 → SATA HDD
    - sata + ROTA=0 → SATA SSD
    - mmc* device → eMMC
    - Else uses transport field capitalized

---

### getDiskModel

- **Description:** Extracts and saves the disk model name.
- **Parameters:**
  - $1 = Disk name
- **Returns:**
  - Echoes and saves model to model.txt
- **Behavior:**
  - Reads columns 7 onward from specifications.txt and joins them into a string

---

### getDiskSize

- **Description:** Extracts and saves the formatted size string (e.g., 512G).
- **Parameters:**
  - $1 = Disk name
- **Returns:**
  - Echoes and saves to size.txt
- **Behavior:**
  - Extracts second column from specifications.txt

### getDiskSerial

- **Description:** Extracts and saves the disk's serial number.
- **Parameters:**
  - $1 = Disk name
- **Returns:**
  - Echoes and saves to serial.txt
- **Behavior:**
  - Extracts third column from specifications.txt

---

### initializeDiskFiles

- **Description:** Creates and resets all necessary tracking files for a given disk.
- **Parameters:**
  - $1 = Disk name
- **Returns:**
  - Creates and zeroes files like type.txt, warnings.txt, health.txt
- **Behavior:**
  - Empties files in the disk’s directory
  - Prepares the reporting structure for that disk

---

### placeSpecsToFolders

- **Description:** Runs all data extraction functions and stores disk metadata into files.
- **Parameters:**
  - $1 = Disk name
- **Returns:**
  - None
- **Behavior:**
  - Calls: getDiskType, getDiskModel, getDiskSize, and getDiskSerial

---

createDiskDir

- **Description:** Reads user-selected disks and initializes directories and files for each.
- **Parameters:**
  - None (relies on CHOSEN_DISKS, CHOSEN_DISKS_COUNT, AUTO_ERASURE_CONF)
- **Returns:**
  - 0 if successfully completed.
  - 1 if unable to create disk directory due to disk not being detected.
  - 2 if unable to initialize disk due to disk name being incorrect.
- **Behavior:**
  - Parses the selected disks line-by-line
  - Validates disk name formatting
  - Creates folder and runs initializeDiskFiles
- **Options:** (depends on AUTO_ERASURE_CONF)
  - **off**: Iterates columns of selected disks
  - **on**: Iterates rows of selected disks

# lib/files/erasureMethods/sata.sh

**Local dependencies:** log.sh

- internal files under lib/files/tmp/

**Description:**
Handles SATA-specific erasure tasks, including secure erase, block erase, overwrite patterns, and handling of HPA/DCO areas. Ensures disk readiness through frozen/locked state checks and includes fallback mechanisms for full data erasure.

---

## compareSectors

- **Description:** Compares two disk sector values to determine if they match.
- **Parameters:**
  - $1 = First sector value
  - $2 = Second sector value
- **Returns:**
  - 0 if sectors are equal.
  - 1 if sectors not equal.
- **Behavior:**
  - Simple equality comparison between two values.

---

## getSataDiskSectors

- **Description:** Retrieves current, maximum, DCO maximum, or combined sector counts from a SATA disk.
- **Parameters:**
  - $1 = Disk name (e.g., sda)
  - $2 = Type of sector info
- **Options ($2):**
  - **current:** Returns currently accessible sectors
  - **max:** Returns max sectors from factory
  - **combined:** Returns current/max as x/y
  - **dco_max:** Returns sectors from DCO configuration
- **Returns:**
  - String output of requested sector info
- **Behavior:**
  - Executes hdparm with different flags and parses output using awk and tr.

---

### checkAndRemoveHPA

- **Description:** Detects and restores HPA to factory default if enabled.
- **Parameters:**
  - $1 = Disk name
- **Returns:**
  - 0: HPA restored
  - 1: HPA restore failed
  - 2: HPA error encountered
  - 3: HPA is not enabled
  - 4: HPA reset skipped
- **Behavior:**
  - Compares current/max sectors and resets HPA using hdparm if needed.
  - Handles errors and logs actions.
  - Triggers disk info refresh if changes are made.
- **Options:**
  - If ERASURE_SPEC_CONF is **NOT** "skip" then performs reset and validation.
  - If ERASURE_SPEC_CONF is "skip" then skips reset.

---

### checkAndRemoveDCO

- **Description:** Attempts to restore disk DCO to expose all sectors.
- **Parameters:**
  - $1 = Disk name
- **Returns:**
  - 0: DCO restored
  - 1: DCO restore failed
  - 2: support not available or unreadable
  - 3: DCO is not enabled
  - 4: DCO reset skipped
- **Behavior:**
  - Compares factory vs DCO max sectors and restores if needed using hdparm --dco-restore. Prompts hibernation if altered.

---

### isDiskFrozen

- **Description:** Detects if the disk is in a frozen state.
- **Parameters:**
  - $1 = Disk name
- **Returns:**
  - Prints yes, no, or UNKNOWN;
  - 0: If not frozen
  - 1: If frozen or state unknown
- **Behavior:**
  - Parses hdparm -I for the frozen state.

---

### isDiskLocked

- **Description:** Detects if the disk is locked under ATA security.
- **Parameters:**
  - $1 = Disk name
- **Returns:**
  - 0: If not locked
  - 1: If locked
- **Behavior:**
  - Uses hdparm -I to search for lock status.

---

### Suspend

- **Description:** Suspends system for a given number of seconds.
- **Parameters:**
  - $1 = Number of seconds
- **Returns:**
  - None
- **Behavior:**
  - Uses rtcwake to put system to sleep and wake automatically.

---

### wakeFromFrozen

- **Description:** Attempts to unfreeze a disk by suspending and resuming the system.
- **Parameters:**
  - $1 = Disk name
- **Returns:**
  - None
- **Behavior:**
  - Runs up to 3 suspend/resume cycles to release the frozen state.

---

### supportsSecureErase

- **Description:** Checks if secure erase is supported on the disk.
- **Parameters:**
- $1 = Disk name
- **Returns:**
  - 0: If supported
  - 1: If not supported
- **Behavior:**
  - Parses hdparm -I output for secure erase capability.

---

### supportsBlockErase

- **Description:** Checks if block erase is supported.
- **Parameters:**
  - $1 = Disk name
- **Returns:**
  - 0: If not supported
  - 1: If supported
- **Behavior:**
  - Parses hdparm -I output for block erase keywords.

---

### secureErase **(Why disk lock check here?)**

- **Description:** Performs ATA Secure Erase using password.
- **Parameters:**
  - $1 = Disk name
- **Returns:**
  - 0: Success
  - 1: Failed
  - 2: Disk locked during erase
- **Behavior:**
  - Sets password and triggers secure erase with status updates. Unlocks disk if locked.

---

### blockErase **(Why disk lock check NOT here?)**

- **Description:** Executes block erase sanitize command on compatible disks.
- **Parameters:**
  - $1 = Disk name
- **Returns:**
  - 0: Success
  - 1: Failed
- **Behavior:**
  - Monitors status using hdparm --sanitize-status and logs progress.

---

### overwriteRandomZero

- **Description:** Overwrites disk with 1 pass of random data followed by 0s.
- **Parameters:**
  - $1 = Disk name
- **Returns:**
  - None
- **Behavior:**
  - Executes shred -n 1 -z -v and writes log to progress file.

---

### overwriteZero

- **Description:** Overwrites disk entirely with zeroes.
- **Parameters:**
  - $1 = Disk name
- **Returns:**
  - None
- **Behavior:**
  - Executes shred -n 0 -z -v and logs to progress file.

---

### isHiddenTriggered

- **Description:** Checks if system should refresh disk info (e.g., after HPA/DCO reset).
- **Parameters:**
  - None
- **Returns:**
  - 0: If refresh is needed
  - 1: If refresh is NOT needed
- **Behavior:**
  - Checks HIDDEN_TRIGGERED environment flag.

# lib/files/erasureMethods/nvme.sh

**Local dependencies:** Writes to: lib/files/tmp/progress/<disk>_progress.txt

**Description:**

Handles NVMe disk information extraction and erasure operations. Supports various secure erase methods including sanitize, crypto sanitize, secure format, and crypto format.

---

## getNvmeDiskSectors

- **Description:** Retrieves the number of sectors (namespace size) from an NVMe device.
- **Parameters:**
  - $1 = Disk name (e.g., nvme0n1)
- **Returns:**
  - Integer sector count
- **Behavior:**
  - Uses nvme id-ns to query namespace size (nsze)
  - Extracts and prints the value as an integer

---

## nvmeSupportsCommand

- **Description:** Checks if a given NVMe erasure method is supported on a disk.
- **Parameters:**
  - $1 = Disk name (e.g., nvme0n1)
  - $2 = Erasure command
- **Options ($2):**
  - **sanitize:** Normal sanitize
  - **cryptoSanitize:** Sanitization with cryptographic erase
  - **secureFormat:** Secure erase with format
  - **cryptoSecureFormat:** Cryptographic format erase
- 
- **Returns:**
  - 0: If method supported.
  - 1: If method NOT supported.
- **Behavior:**
  - Parses controller fields (sanicap and fna)
  - Uses bitwise checks to determine command support
  - Exports NVME_SUPPORTS_COMMAND with result message

---

### nvmeCryptoSanitize

- **Description:** Executes a cryptographic NVMe sanitize operation.
- **Parameters:**
  - $1 = Disk name (e.g., nvme0n1)
- **Returns:**
  - 0: Success
  - 1: Failure
- **Behavior:**
  - Issues nvme sanitize -a 4
  - Continuously polls nvme sanitize-log to monitor status
  - Logs progress to temporary file

---

### nvmeSanitize

- **Description:** Executes a standard NVMe sanitize operation.
- **Parameters:**
  - $1 = Disk name
- **Returns:**
  - 0: Success
  - 1: Failure
- **Behavior:**
  - Issues nvme sanitize -a 2
  - Polls nvme sanitize-log until sanitize completes
  - Writes feedback to progress file

---

### nvmeFormatSecure

- **Description:** Performs a secure NVMe format using secure erase (-s 1).
- **Parameters:**
  - $1 = Disk name
- **Returns:**
  - 0: Success
  - 1: Failure
- **Behavior:**
  - Runs nvme format -s 1 –force
  - Logs result to progress file

---

### nvmeFormatCrypto

- **Description:** Performs a cryptographic NVMe format (-s 2).
- **Parameters:**
  - $1 = Disk name
- **Returns:**
  - 0: Success
  - 1: Failure
- **Behavior:**
  - Runs nvme format -s 2 –force
  - Logs result to progress file

# lib/files/erasureMethods/emmc.sh

**Local dependencies:** TMP_PROGRESS file for status updates

**Description:**
Provides secure erasure and trimming methods for eMMC drives using standard mmc commands. It includes functionality for sanitize, secure erase, secure trim, trim, discard, and legacy erase, each handling progress reporting and return codes.

---

## getEmmcDiskSectors

- **Description:** Retrieves the total number of addressable sectors for a given eMMC drive.
- **Parameters:**
  - $1 = Disk name (e.g., mmcblk0)
- **Returns**
  - Integer representing total sectors
- **Behavior:**
  - Reads /sys/block/<disk>/size to determine the number of sectors on the disk.

---

## getEmmcSectorSize

- **Description:** Retrieves the hardware sector size for a given eMMC drive.
- **Parameters:**
  - $1 = Disk name
- **Returns:**
  - Integer representing sector size in bytes
- **Behavior:**
  - Reads /sys/block/<disk>/queue/hw_sector_size to get the drive's hardware sector size.

---

## mmcSanitize

- **Description:** Performs an MMC sanitize operation on the specified eMMC disk.
- **Parameters:**
  - $1 = Disk name
- **Returns:**
  - 0: Success
  - 1: Failure
- **Behavior:**
  - Runs mmc sanitize /dev/<disk> in the background, waits for it to complete, logs the result to TMP_PROGRESS.

---

### mmcSecureErase

- **Description:** Performs a secure erase across all sectors of an eMMC drive.
- **Parameters:**
  - $1 = Disk name
- **Returns:**
  - 0: Success
  - 1: Failure
- **Behavior:**
  - Uses mmc erase secure-erase 0 <total sectors> on the specified disk. Waits for completion and logs result.

---

### mmcSecureTrim1

- **Description:** Performs a secure trim (method 1) on the eMMC disk.
- **Parameters:**
  - $1 = Disk name
- **Returns:**
  - 0: Success
  - 1: Failure
- **Behavior:**
  - Uses mmc erase secure-trim1 on the full range of sectors. Waits and logs outcome.

---

### mmcSecureTrim2

- **Description:** Performs a secure trim (method 2) on the eMMC disk.
- **Parameters:**
  - $1 = Disk name
- **Returns:**
  - 0: Success
  - 1: Failure
- **Behavior:**
  - Uses mmc erase secure-trim2 across all sectors, waits and logs results.

---

### mmcTrim

- **Description:** Performs a non-secure trim operation on the entire eMMC disk.
- **Parameters:**
  - $1 = Disk name
- **Returns:**
  - 0: Success
  - 1: Failure
- **Behavior:**
  - Executes mmc erase trim 0 <total sectors>, waits for the operation to complete, and logs the result.

---

### mmcDiscard

- **Description:** Executes a discard operation across the eMMC disk.
- **Parameters:**
  - $1 = Disk name
- **Returns:**
  - 0: Success
  - 1: Failure
- **Behavior:**
  - Runs mmc erase discard 0 <total sectors> to discard data and logs the outcome.

---

### mmcLegacy

- **Description:** Performs a legacy erase operation on the eMMC drive.
- **Parameters:**
  - $1 = Disk name
- **Returns:**
  - 0: Success
  - 1: Failure
- **Behavior:**
  - Runs mmc erase legacy 0 <total sectors> and logs the result to the TMP_PROGRESS file.

# lib/verifyErasure.sh

**Local dependencies:** getDiskSizeInBytes, overwriteZero

**Description:**
Verifies that a disk has been fully wiped by scanning for non-zero data patterns using several verification modes. If non-zero data is found, the verification fails.

---

## verifyErasure

- **Description:** Verifies that the specified disk contains only zero data. The mode of verification determines how much of the disk is scanned and how the results are reported.
- **Parameters:**
  - $1 = Disk name (e.g., sda, nvme0n1)
  - $2 = Block size to use with dd (e.g., 1M, 64M)
  - $3 = Verification type (e.g., full, partial, skip, quick_check, snapshot_before, snapshot_after, compare_snapshots)
- **Options ($3):**
  - **full**: Scans the entire disk for non-zero bits.
  - **partial**: Scans the first and last 10% of the disk.
  - **quick_check**: Scans just the first few blocks to quickly detect patterns.
  - **skip**: Skips verification entirely and logs it.
  - **snapshot_before**: Takes a snapshot before crypto erase.
  - **snapshot_after**: Takes a snapshot after crypto erase.
  - **compare_snapshots**: Compares before/after snapshots for changes using SHA256.
- **Returns:**
  - 0: Success (disk verified, no non-zero bits found)
  - 1: Failure (non-zero bits found)
  - 2: Skipped
  - 4: Snapshot comparison fail
  - 5: Snapshot comparison success
  - 6: Quick check success
  - 7: Quick check fail
- **Behavior:**
  - For full, reads entire disk and scans for non-zero bits.
  - For partial, reads the first and last 10% of the disk.
  - For quick_check, reads only the first two blocks.
  - For skip, writes SKIPPED to logs and exits early.
  - For snapshot_before and snapshot_after, saves hex snapshots.
  - For compare_snapshots, compares SHA256 hashes of before/after snapshots.
  - Logs all output and status to structured progress and result files.

---

### quickCheckAndOverwrite

- **Description:** Checks if the disk contains a zero pattern using a quick check. If not, optionally overwrites the disk with zeros and logs the action.
- **Parameters:**
  - $1 = Disk name
- **Returns:**
  - None
- **Behavior:**
  - Controlled by the config variable CHECK_ZERO_PATTERN_AND_OVERWRITE_CONF.
  - If set to "on": performs quick check and overwrites if non-zero bits are found.
  - If set to "off": warns user if non-zero bits are found, but does not overwrite.
  - Uses the verifyErasure function in quick_check mode to detect data.

## lib/files/erasureSpecs/sata_ssd.sh

**Local dependencies:**

- secureErase, overwriteZero, overwriteRandomZero, blockErase, supportsSecureErase
- Writes to: lib/files/tmp/chosenDisks/<disk>/method.txt, tool.txt, spec.txt, warnings.txt

**Description:**
Implements secure erasure methods for SATA SSDs. Supports both "Purge" and "Clear" level erasure specs through combinations of block erase, secure erase, and overwriting. Used to meet NIST requirements.

---

### sata_ssd_purge_block_erase

- **Description:** Performs a "Purge" level erasure on an SSD using the sequence: Block Erase → Secure Erase/Zero Fill → Block Erase.
- **Parameters:**
  - $1 = Disk name (e.g., sda)
- **Returns:**
  - 0 = Success (first block erase completed)
  - 1 = Failure (first block erase failed)
- **Behavior:**
  - Calls supportsSecureErase to determine drive capabilities.
  - Performs two attempts of block erase as the first step.
  - If the first erase succeeds, continues to optional secure erase (or zero fill if unsupported).
  - Finishes with a second block erase.
  - Logs methods, tools used, and spec ("Purge") to appropriate files.
  - Any failure is noted and stored in warnings.txt.

---

### sata_ssd_clear_secure_erase

- **Description:** Performs a "Clear" level secure erase using hdparm's secure erase functionality.
- **Parameters:**
  - $1 = Disk name (e.g., sda)
- **Returns:**
  - 0: Secure erase succeeded
  - 1: Secure erase failed
- **Behavior:**
  - Calls secureErase to run firmware secure erase.
  - Appends outcome to method.txt, tool version to tool.txt, and spec.txt as "Clear".
  - Writes result as either successful or failed based on return code.

---

### sata_ssd_clear_overwrite

- **Description:** Performs a "Clear" level erasure using overwrite operations, either zero fill or random + zero.
- **Parameters:**
  - $1 = Disk name (e.g., sda)
  - $2 = Type of overwrite (one, two)
- **Options ($2):**
  - **one**: Overwrites entire disk with zeros.
  - **two**: Overwrites with random data, then with zeros for stronger "clear" assurance.
- **Returns:**
  - None
- **Behavior:**
  - If type is "one", runs overwriteZero (zero fill).
  - If type is "two", runs overwriteRandomZero (random pattern followed by zero fill).
  - Appends result to method.txt, records shred version, and writes "Clear" to spec.txt.

## lib/files/erasureSpecs/sata_hdd.sh

**Local dependencies:**

- secureErase, overwriteZero, overwriteRandomZero
- Writes to: lib/files/tmp/chosenDisks/<disk>/method.txt, tool.txt, spec.txt

**Description:**

Implements secure data erasure methods for SATA HDDs according to NIST 800-88 recommendations. Provides both "Purge" and "Clear" level erasure options using firmware-based secure erase and software-based overwrite strategies.

---

### sata_hdd_purge_secure_erase

- **Description:** Executes a firmware-based secure erase on a SATA HDD to meet "Purge" level data sanitization as per NIST 800-88.
- **Parameters:**
  - $1 = Disk name (e.g., sda)
- **Returns:**
  - 0: Secure erase succeeded
  - 1: Secure erase failed
- **Behavior:**
  - Checks if prior methods exist; adds " >" to methodMessage if so.
  - Calls the secureErase function.
  - Logs tool version from hdparm.
  - Records method, tool, and spec (Purge) into disk report directory.
  - Returns appropriate status based on the success of the secure erase.

---

### sata_hdd_clear_overwrite

- **Description:** Performs software-based overwrite of the SATA HDD using either zero-fill or a two-pass random-then-zero method to meet "Clear" level erasure standards.
- **Parameters:**
  - $1 = Disk name (e.g., sda)
  - $2 = Overwrite type: one for zero fill, two for random + zero
- **Options ($2):**
  - **one**: Overwrites entire disk with zeros.
  - **two**: Overwrites with random data followed by zeros for added assurance.
- **Returns:**
  - None
- **Behavior:**
  - Adds " >" to methodMessage if other erasure attempts were logged.
  - If type is "one": runs overwriteZero.
  - If type is "two": runs overwriteRandomZero.
  - Logs the shred tool version.
  - Appends method, tool, and "Clear" spec info to report files.

## lib/files/erasureSpecs/nvme.sh

**Local dependencies:**

- verifyErasure, nvmeCryptoSanitize, nvmeFormatCrypto, nvmeSanitize, nvmeSupportsCommand, nvmeFormatSecure, overwriteZero, overwriteRandomZero

**Description:**

Handles erasure of NVMe SSDs using NIST-compliant methods. Implements a cascade of secure operations—Crypto Erase, Sanitize, Secure Format, and Overwrite—based on device capability. Supports both "Purge" and "Clear" levels.

---

### nvme_purge_crypto_erase

- **Description:** Performs Crypto Sanitize, followed by Sanitize or Secure Format if available. Falls back to zero-overwrite if those fail.
- **Parameters:**
  - $1 = Disk name (e.g., nvme0n1)
- **Returns:**
  - 0: Success
  - 1: Failure
- **Behavior:**
  - Takes pre-wipe snapshot of disk and performs Crypto Sanitize.
  - If Crypto Sanitize succeeds, it compares hash snapshots to confirm wipe.
  - If verified, tries Sanitize → Secure Format → Overwrite [Zero], in that order.
  - If Crypto Sanitize fails or verification fails, notes as [FAILED].
  - Logs method chain, tool version(s), and spec (Purge) to disk-specific report files.

---

### nvme_purge_crypto_format

- **Description:** Runs Crypto Format (NVMe firmware-based erase) and follows similar fallback steps as above: Sanitize → Secure Format → Overwrite [Zero].
- **Parameters:**
  - $1 = Disk name (e.g., nvme0n1)
- **Returns:**
  - 0: Success
  - 1: Failure
- **Behavior:**
  - Performs Crypto Format and captures snapshots before and after.
  - Validates via hash comparison.
  - On success, continues with Sanitize or Secure Format, or falls back to Overwrite.
  - Method chain is recorded in method.txt, tool.txt, and spec.txt.

---

### nvme_purge_sanitize

- **Description:** Runs the sanitize NVMe command. If successful or if secure format is supported, attempts that too.
- **Parameters:**
  - $1 = Disk name (e.g., nvme0n1)
- **Returns:**
  - 0: At least one method succeeded
  - 1: Both sanitize and format failed
- **Behavior:**
  - Executes Sanitize command.
  - If Secure Format is available, tries that too.
  - Logs results and tool versions under the "Purge" spec.

---

### nvme_purge_format

- **Description:** Performs a Secure Format command for "Purge" level erasure.
- **Parameters:**
  - $1 = Disk name (e.g., nvme0n1)
- **Returns:**
  - 0: Success
  - 1: Failure
- **Behavior:**
  - Runs nvmeFormatSecure and logs result.
  - Uses nvme --version to log tool version.
  - Writes method and spec information to report files.

---

### nvme_clear_overwrite

- **Description:** Performs "Clear" level overwrite using zero or random-zero passes.
- **Parameters:**
  - $1 = Disk name
  - $2 = Overwrite type (one, two)
- **Options ($2):**
  - **one**: Overwrites the entire disk with zeros.
  - **two**: Overwrites with random data followed by zeros for additional security.
- **Returns:**
  - None
- **Behavior:**
  - If type is one, runs overwriteZero.
  - If type is two, runs overwriteRandomZero.
  - Logs overwrite method, shred version, and "Clear" spec.

## lib/files/erasureSpecs/emmc.sh

**Local dependencies:**

- mmcSanitize, mmcSecureErase, overwriteZero, overwriteRandomZero

**Description:**

Handles secure data erasure for eMMC storage using mmc-utils commands. Attempts Sanitize and Secure Erase, with optional overwrite fallbacks to meet "Clear" level data erasure requirements.

---

### emmc_clear

- **Description:** Performs eMMC disk sanitization using mmc-utils and optionally applies one or two-pass software overwrites for added security. Logs all actions and tool versions to report files for auditability.
- **Parameters:**
  - $1 = Disk name (e.g., mmcblk0)
  - $2 = Overwrite type: one or two
- **Options:**
  - **one**: Appends a single zero-pass overwrite using overwriteZero.
  - **two**: Applies a two-pass overwrite (random followed by zero) using overwriteRandomZero.
- **Returns:**
  - None
- **Behavior:**
  - Calls mmcSanitize and mmcSecureErase, logging their exit codes.
  - Appends result summary to method.txt for each command (notes [FAILED] if unsuccessful).
  - Based on $2, optionally performs overwrite:
    - "one" triggers overwriteZero
    - "two" triggers overwriteRandomZero
  - Logs tool versions (mmc-utils, shred) in tool.txt
  - Marks the erasure spec as "Clear" in spec.txt

## lib/files/erasureSpecs/not_supported.sh

**Local dependencies:**

- overwriteZero, overwriteRandomZero
- Writes to: lib/files/tmp/chosenDisks/<disk>/method.txt, tool.txt, spec.txt

**Description:**

Provides a fallback erasure mechanism for drives that do not support firmware-based secure erase commands. Implements basic overwrite routines aligned with NIST "Clear" level sanitization.

---

### not_supported_clear

- **Description:** Executes a software-based data erasure for unsupported disks using one-pass zero fill or two-pass random and zero fill, then logs metadata for auditing.
- **Parameters:**
  - $1 = Disk name (e.g., sdb, nvme1n1)
  - $2 = Overwrite type (one, two)
- **Options ($2):**
  - **one**: Overwrites the entire disk with zeros.
  - **two**: Performs a random overwrite followed by a zero pass for added assurance.
- **Returns:**
  - None
- **Behavior:**
  - Checks if the method log already exists and appends an arrow ( >) if so.
  - Depending on the type:
  - "one" runs overwriteZero, logging [Zero]
  - "two" runs overwriteRandomZero, logging [Random > Zero]
  - Logs the version of shred used.
  - Appends method, tool, and "Clear" spec to report files under the disk path.

# lib/erasureAlgorithm.sh

**Local dependencies:**

- erasureSpecs/sata_ssd_*, erasureSpecs/sata_hdd_*, erasureSpecs/nvme_*, erasureSpecs/emmc_clear, erasureSpecs/not_supported_clear, quickCheckAndOverwrite, erasureMethods/*/supports*
- Output files: method.txt, tool.txt, spec.txt, warnings.txt

**Description:**

Determines and executes the correct erasure algorithm for a given disk based on its type and configured erasure standard (purge, clear, auto, skip). Logs results for auditability and compliance.

---

## erasureSkip

- **Description:** Logs that erasure has been skipped for a given disk and marks method/tool/spec as N/A.
- **Parameters:**
  - $1 = Disk name (e.g., sda)
- **Returns:**
  - None
- **Behavior:**
  - Writes "N/A" values to method, tool, and spec report files.
  - Appends a message to warnings.txt.

---

## Erasure

- **Description:** Main function for handling disk erasure logic. Determines the erasure path based on disk type and desired standard (purge, clear, auto, skip).
- **Parameters:**
  - $1 = Disk name (e.g., sda, nvme0n1)
  - $2 = Erasure standard (purge, clear, auto, skip)
- **Options ($2):**
  - **purge**: Attempts highest level secure erasure supported by hardware (e.g., crypto erase, sanitize). If unsupported, logs warning and skips.
  - **clear**: Applies firmware-based secure erase if supported, else falls back to two-pass software overwrite.
  - **auto**: Tries purge first; if unsupported, attempts clear. Ensures fallback coverage.
  - **skip**: No erasure performed. Logs "N/A" in all relevant output fields.
- **Returns:**
  - None
- **Behavior:**
  - Reads disk type from type.txt and selects appropriate secure erasure method.
  - Logs warnings if configured verification level is insufficient. Executes erasure based on disk type:
    - **SATA SSD:** Uses block erase, secure erase, or overwrite with quickCheckAndOverwrite
    - **SATA HDD:** Uses secure erase or overwrite
    - **NVME:** Tries several secure NVMe erase commands (sanitize, crypto format, etc.)
    - **eMMC/MMC:** Uses emmc_clear with specified overwrite type
    - **Unsupported types:** Falls back to not_supported_clear
    - Skips erasure entirely if standard is skip

## lib/erasureProgress.sh

**Local dependencies:**

- Reads from:
    - lib/files/tmp/progress/*_progress.txt
    - lib/files/tmp/chosenDisks/<disk>/size.txt
    - lib/files/tmp/chosenDisks/<disk>/type.txt

**Description:**

Displays progress of disk erasure either through a terminal-based display or a Whiptail GUI popup. Retrieves status from individual progress text files created during erasure and prints human-readable progress reports for each disk.

---

### erasureProgressTUI

- **Description:** Displays progress of each disk erasure using Whiptail TUI. Requires manual refresh by pressing "OK."
- **Parameters:**
    - None
- **Returns:**
    - None
- **Behavior:**
    - Loops through progress files (*_progress.txt) in the temp directory.
    - Extracts disk name, type, and size from relevant files.
    - If the progress file doesn't exist, shows "Starting erasure.."
    - If the file exists, cleans it of null characters using tr -d '\000' and formats it using awk.
    - Appends all progress messages into one string and displays it using whiptail --msgbox.
    - Requires manual confirmation (OK) to refresh.

---

### erasureProgress

- **Description:** Displays progress of disk erasure directly in the terminal.
- **Parameters:**
    - None
- **Returns:**
    - None
- **Behavior:**
    - Loops through progress files (*_progress.txt) in the temp directory.
    - Extracts disk name, type, and size from relevant files.
    - If a progress file exists, sanitizes it and displays the contents; otherwise shows a startup message.
    - Builds and echoes a formatted summary for all disks being erased.
    - Clears the progress message buffer after each update.

# lib/diskHealth.sh

**Local dependencies:**

- SMART_TEST_CONF (environment variable set elsewhere: short, long, skip)
- Output directories: lib/files/tmp/chosenDisks/<disk>/health.txt, lib/files/tmp/progress/<disk>_progress.txt, verification.txt

**Description:**

Handles S.M.A.R.T. disk health tests and logs the results. Supports both short and long self-tests depending on configuration, and manages test progress reporting during runtime.

---

## healthSupported

- **Description:**
  - Checks if the selected disk supports S.M.A.R.T. testing.
- **Parameters:**
  - $1 = Disk name (e.g., sda, nvme0n1)
- **Returns:**
  - 0: Supported
  - 1: Not supported
- **Behavior:**
  - Runs smartctl -H to detect S.M.A.R.T. support.
  - Parses output for errors like "Unable to detect device type".
  - Logs "Health: Not Supported" if not supported.

---

## diskHealthResult

- **Description:**
  - Extracts and logs the result of the most recent S.M.A.R.T. test.
- **Parameters:**
  - $1 = Disk name (e.g., sda, nvme0n1)
- **Returns:**
  - None
- **Behavior:**
  - Appends the S.M.A.R.T. test result line (containing "result") to health.txt.

---

### checkHealthProgress

- **Description:** Tracks and logs progress of an ongoing S.M.A.R.T. test to the UI progress file.
- **Parameters:**
  - $1 = Disk name
- **Returns:**
  - None
- **Behavior:**
  - Periodically checks for "remaining" percentage using smartctl -c.
  - Detects estimated duration of the current test (short or long).
  - Updates progress.txt file every 2 seconds with current status.
  - Ends loop once the percentage is no longer available.

---

### startTest

- **Description:** Begins a S.M.A.R.T. test and invokes progress tracking.
- **Parameters:**
  - $1 = Disk name
  - $2 = Test type (short or long)
- **Options:**
  - **short**: Standard quick test, usually under 2 minutes.
  - **long**: Extended, more comprehensive scan (can take hours).
- **Returns:**
  - None
- **Behavior:**
  - Launches S.M.A.R.T. test via smartctl -t.
  - Waits 2 seconds, then calls checkHealthProgress.

---

### diskHealth

- **Description:** Main entry point for checking and logging S.M.A.R.T. test results.
- **Parameters:**
  - $1 = Disk name
- **Returns:**
  - None
- **Behavior:**
  - If SMART_TEST_CONF is not skip, runs healthSupported.
  - If supported, runs startTest, saves result, and updates progress.
  - If not supported, appends verification message to progress file.
  - If set to skip, logs "status: Skipped" to health.txt.
- **Options (via SMART_TEST_CONF):**
  - **short**: Runs quick S.M.A.R.T. test
  - **long**: Runs extended S.M.A.R.T. test
  - **skip**: Skips health testing entirely and marks as skipped

## lib/taskManager.sh

**Local dependencies:**

- isDiskFrozen, wakeFromFrozen, getSataDiskSectors, getDiskSectors, getDiskSize, getFullDiskSpecs, checkAndRemoveHPA, checkAndRemoveDCO, erasure, verifyErasure, diskHealth, log, erasureProgress

**Description:**
This is the central task orchestrator for disk processing. It handles HPA/DCO removal, erasure, verification, and S.M.A.R.T. health checks in parallel using process tracking. It ensures correct ordering and captures results and metadata per disk.

---

### TaskManager

- **Description:** Manages the lifecycle of selected disk erasure. Ensures frozen state handling, secure wipe initiation, progress monitoring, post-wipe verification, and health checking — all asynchronously for multiple disks.
- **Parameters:**
  - None (invoked globally, operates on all chosenDisks)
- **Options:** Not directly passed but relies on global configurations such as:
  - ERASURE_SPEC_CONF: Determines erasure mode (e.g., overwrite, secure erase)
  - VERIFICATION_CONF: Determines verification depth (full, partial, skip)
- **Returns:**
  - None
- **Behavior:**
  - Iterates over each selected disk in chosenDisks/.
  - Detects frozen state for SATA drives and attempts to unfreeze.
  - Logs the start time and extracts initial sector data.
  - Removes HPA and DCO where applicable and refreshes disk metadata.
  - Begins erasure in the background and tracks PIDs in disk_erasure_pids.
  - Continuously checks for completion of each erasure process.
  - When erasure completes, logs the result and launches verification.
  - After verification, launches a background S.M.A.R.T. health check.
  - When all disks are processed (erasure + verify + health), it compiles status messages and displays a summary to the user via whiptail.

## lib/reportGenerator.sh

**Local dependencies:**

- ifReportExists
- Files/folders used:
    - lib/files/tmp/chosenDisks/
    - lib/files/tmp/verificationStatus/
    - lib/files/tmp/attachedDisks.txt
    - lib/files/stylesheet/reportStyle.xsl
    - lib/files/reports/

**Description:**

Generates a complete erasure report in XML and PDF format. Includes metadata (e.g., asset tag, BIOS time, erasure configuration), detailed disk information (e.g., model, serial, tool used, erasure method, verification results), and full system specifications. Uses XSLT and Apache FOP to produce a styled PDF report.

---

### reportToPDF

- **Description:** Converts the generated XML report into a PDF using a custom XSLT stylesheet and Apache FOP. Stylesheet is at lib/files/stylesheet/reportStyle.xsl.
- **Parameters:**
    - None
- **Returns:**
    - Creates lib/files/reports/<ASSET_TAG>.pdf and deletes the temporary .fo file.
- **Behavior:**
    - Computes SHA-256 hash of the XML.
    - Injects metadata and configuration into XSLT processor.
    - Applies the stylesheet to format XML into XSL-FO (tmp.fo).
    - Uses FOP to convert to PDF.

---

### insertDiskInfo

- **Description:** Appends detailed disk information to the report XML file. Each disk block contains erasure details, tool used, and verification outcome.
- **Parameters:**
  - None
- **Returns:**
  - Appends <disks>...</disks> block into $REPORT.
- **Behavior:**
  - Iterates through lib/files/tmp/chosenDisks/
  - Extracts:
    - Serial number, model, size, type, HPA/DCO status
    - Erasure method, tool, time, sectors before/after
    - Health and verification results
    - Handles special cases:
      - UNKNOWN model fallback
      - Type fix for mmc
      - Extra tool parsing for NVMe

---

### reportGenerator

- **Description:** Main function for generating full XML report, which includes asset tag, metadata, disk info, system specs, and PDF conversion.
- **Parameters:**
  - None
- **Returns:**
  - Creates lib/files/reports/<ASSET_TAG>.xml and .pdf
- **Behavior:**
  - Creates the XML root and adds metadata like technician, customer, etc.
  - Checks disk presence:
    - If none: logs status (not detected, canceled, etc.)
    - If disks present: adds warnings, boot disk info, inserts disk data
  - Appends system specs using lshw
  - Calls reportToPDF

---

### hashOfXML

- **Description:**
  - Generates and stores SHA-256 hash of the completed XML report.
- **Parameters:**
  - None
- **Returns:**
  - None
- **Behavior:**
  - Calculates hash from lib/files/reports/<ASSET_TAG>.xml
  - Stores it in HASH for use in reportToPDF

# Template:

## ScriptName.sh

**Local dependencies:** something something

**Description:**

something something

### MethodName

- **Description:** somethingsomething
- **Parameters:**
  $1= something
- **Options:**
  - Option: Something
  - Option: somethingsomething
- **Returns:**
  - something something
- **Behavior:**
  - something something