Active@ KillDisk

USER MANUAL

ver. 26.0

Published: 13 Mar 2026

Contents

Legal Statement

Introduction

As a relatively new technology an overwhelming majority of people, businesses and organizations do not understand the importance of security in digital data storage. The average hard drive stores thousands of files written on it and many of them contain sensitive information. Over the course of a hard drives lifetime the likelihood for recoverable remnants of sensitive information left on a hard drive at its end of life is very high. To see this just try out **Active@ KillDisk**'s My Computer view on your system drive. You'll be surprised to see what you find!

The modern storage environment is rapidly evolving. Data may pass through multiple organizations, systems, and storage media in its lifetime. The pervasive nature of data propagation is only increasing as the Internet and data storage systems move towards a distributed cloud-based architecture. As a result, more parties than ever are responsible for effectively sanitizing media and the potential is substantial for sensitive data to be collected and retained on the media. This responsibility is not limited to those organizations that are the originators or final resting places of sensitive data, but also intermediaries who transiently store or process the information along the way. The efficient and effective management of information from inception through disposition is the responsibility of all those who have handled the data.

The application of sophisticated access controls and encryption help reduce the likelihood that an attacker can gain direct access to sensitive information. As a result, parties attempting to obtain sensitive information may seek to focus their efforts on alternative access means such as retrieving residual data on media that has left an organization without sufficient sanitization effort having been applied. Consequently, the application of effective sanitization techniques and tracking of storage media are critical aspects of ensuring that sensitive data is effectively protected by an organization against unauthorized disclosure. Protection of information is paramount. That information may be on paper, optical, electronic or magnetic media.

An organization may choose to dispose of media by charitable donation, internal or external transfer, or by recycling it in accordance with applicable laws and regulations if the media is obsolete or no longer usable.

Even internal transfers require increased scrutiny, as legal and ethical obligations make it more important than ever to protect data such as Personally Identifiable Information (PII). No matter what the final intended destination of the media is, it is important that the organization ensure that no easily recoverable residual representation of the data is stored on the media after it has left the control of the organization or is no longer going to be protected at the confidentiality categorization of the data stored on the media.

Sanitization refers to a process that renders access to target data on the media infeasible for a given level of effort.

Note:

Additionally, try formatting a USB drive with files on it and browse it with **Active@ KillDisk**'s My Computer view as well. Data leakages are not limited to hard drives!

Sanitization Types

NIST 800-88 international security standard (Guidelines for Media Sanitization) defines different types of sanitization.

Regarding sanitization, the principal concern is ensuring that data is not unintentionally released. Data is stored on media, which is connected to a system. Simply data sanitization applied to a representation of the data as stored on a specific media type.

When media is re-purposed or reaches end of life, the organization executes the system life cycle sanitization decision for the information on the media. For example, a mass-produced commercial software program contained on a DVD in an unopened package is unlikely to contain confidential data. Therefore, the decision may be made to simply dispose of the media without applying any sanitization technique. Alternatively, an organization is substantially more likely to decide that a hard drive from a system that processed Personally Identifiable Information (PII) needs sanitization prior to Disposal.

Disposal without sanitization should be considered only if information disclosure would have no impact on organizational mission, would not result in damage to organizational assets, and would not result in financial loss or harm to any individuals. The security categorization of the information, along with internal environmental factors, should drive the decisions on how to deal with the media. The key is to first think in terms of information confidentiality, then apply considerations based on media type. In organizations, information exists that is not associated with any categorized system. Sanitization is a process to render access to target data (the data subject to the sanitization technique) on the media infeasible for a given level of recovery effort. The level of effort applied when attempting to retrieve data may range widely. NIST SP 800-88 Rev. 1 Guidelines for Media Sanitization *Clear*, *Purge*, and *Destroy* are actions that can be taken to sanitize media. The categories of sanitization are defined as follows:

Clear

Clear applies logical techniques to sanitize data in all user-addressable storage locations for protection against simple non-invasive data recovery techniques; typically applied through the standard Read and Write commands to the storage device, such as by rewriting with a new value or using a menu option to reset the device to the factory state (where rewriting is not supported).

For HDD/SSD/SCSI/USB media this means overwrite media by using organizationally approved and validated overwriting technologies/methods/tools. *The Clear pattern* should be at least a single write pass with a fixed data value, such as all zeros. Multiple write passes or more complex values may optionally be used.

Active@ KillDisk supports *Clear* sanitization type through the ***Disk Erase*** command for all R/W magnetic types of media, more than 20 international sanitation methods including custom patterns implemented and can be used.

Purge

Purge applies physical or logical techniques that render Target Data recovery infeasible using state of the art laboratory techniques.

For HDD/SSD/SCSI/USB media this means ATA SECURE ERASE UNIT, ATA CRYPTO SCRAMBLE EXT, ATA EXT OVERWRITE, ATA/SCSI SANITIZE and other low-level direct controller commands.

Active@ KillDisk supports *Purge* sanitization type through the *Secure Erase* command only for media types supporting ATA extensions.

Destroy

Destroy renders Target Data recovery infeasible using state of the art laboratory techniques and results in the subsequent inability to use the media for storage of data due to physical damages.

For HDD/SSD/SCSI media this means Shred, Disintegrate, Pulverize, or Incinerate by burning the device in a licensed incinerator.

It is suggested that the user categorize the information, assess the nature of the medium on which it is recorded, assess the risk to confidentiality, and determine the future plans for the media. Then, the organization can choose the appropriate type(s) of sanitization. The selected type(s) should be assessed as to cost, environmental impact, etc., and a decision should be made that best mitigates the risk to confidentiality and best satisfies other constraints imposed on the process.

International standards in data destruction

Sanitizing methods

Active@ KillDisk works with dozens of international sanitizing methods for clearing and sanitizing data including the *US DoD 5220.22-M* and *NIST 800-88* standards. You can be sure that once you erase a disk with **Active@ KillDisk** all the sensitive information is destroyed forever.

Active@ KillDisk is a professional security application that destroys data permanently from any computer that can be started using a boot USB or CD/DVD. Access to the drive's data is made on the physical level via the *BIOS Settings* (Basic Input-Output Subsystem) bypassing the operating system's logical drive structure organization. Regardless of the operating system, file systems or machine types, this utility can destroy all data on all storage devices. It does not matter which operating systems or file systems are located on the machine which disks being sanitized.

Active@ KillDisk supports the following international sanitizing methods:

- *One Pass Zeros or One Pass Random*
- *US DoD 5220.22-M*
- *US DoE M205.1-2*
- *Canadian CSEC ITSG-06*
- *Canadian OPS-II*
- *British HMG IS5 Baseline*>
- *British HMG IS5 Enhanced*
- *Russian GOST p50739-95*
- *US Army AR380-19*
- *US Air Force 5020*
- *NAVSO P-5329-26 RL*
- *NCSC-TG-025*
- *NSA 130-2*
- *NIST 800-88*
- *German VSITR*

- *Bruce Schneier*
- *Peter Gutmann*
- *Australian ISM-6.2.93*
- *IEEE Std 2883-2022*
- *User Defined*

User Defined erase method

Active@ KillDisk offers *User Defined* erase method where user indicates the number of times the write head passes over each sector. Each overwriting pass is performed with a buffer containing user-defined or random characters. User Defined method allows to define any kind of new erase algorithms based on user requirements.

Secure Erase for SSD

Active@ KillDisk offers low-level ATA *Secure Erase (ANSI ATA, SE)* method for Solid State Drives (SSD). According to National Institute of Standards and Technology (NIST) Special Publication 800-88: Guidelines for Media Sanitation, *Secure Erase* is "*An overwrite technology using firmware based process to overwrite a hard drive. Is a drive command defined in the ANSI ATA and SCSI disk drive interface specifications, which runs inside drive hardware. It completes in about 1/8 the time of 5220 block erasure.*" The guidelines also state that "*degaussing and executing the firmware Secure Erase command (for ATA drives only) are acceptable methods for purging.*" ATA Secure Erase (SE) is designed for SSD controllers. The SSD controller resets all memory cells making them empty. In fact, this method restores the SSD to the factory state, not only deleting data but also returning the original performance. When implemented correctly, this standard processes all memory, including service areas and protected sectors.

Related concepts

US DoD 5220.22-M erase method on page 188
The **U.S. Department of Defense (DoD) 5220.22-M** erase method is a **data sanitization standard** historically defined in the **National Industrial Security Program Operating Manual (NISPOM)**. It specifies a process for **securely overwriting data on magnetic storage media** (e.g., hard drives, tapes) to prevent recovery of sensitive information.

Canadian CSEC ITSG-06 erase method on page 189
The **Canadian CSEC ITSG-06** (Communications Security Establishment Canada – *Information Technology Security Guidance No. 6*) erase method is a **data sanitization guideline** issued by the **CSEC**, which governs how to securely erase information from electronic storage devices to prevent unauthorized data recovery.

Related information

Erase Methods on page 186

Erase Confidential Data

Modern methods of data encryption are deterring network attackers from extracting sensitive data from stored database files.

Attackers (who want to retrieve confidential data) become more resourceful and look for places where data might be stored temporarily. For example, the Windows ***DELETE*** command merely changes the files attributes and location so that the operating system will not look for the file located on FAT/exFAT volumes. The situation with NTFS file system is similar.

One avenue of attack is the recovery of data from residual data on a discarded hard drive. When deleting confidential data from hard drives, removable disks or USB devices, it is important to extract all traces of the data so that recovery is not possible.

Most official guidelines regarding the disposal of confidential magnetic data do not take into account the depth of today's recording densities nor the methods used by the OS when removing data.

Removal of confidential personal information or company trade secrets in the past might have been performed using the ***FORMAT*** command or the ***FDISK*** command. Using these procedures gives users a sense of confidence that the data has been completely removed.

When using the ***FORMAT*** command Windows displays a message like this: `Formatting a disk removes all information from the disk.`

Actually the ***FORMAT*** utility creates new empty directories at the root area, leaving all previous data on the disk untouched. Moreover, an image of the replaced FAT tables is stored so that the ***UNFORMAT*** command can be used to restore them.

FDISK merely cleans the Partition Table (located in the drive's first sector) and does not touch anything else.

Related concepts

Erase disk concept on page 184
Understanding of underlying mechanisms of data storage organization and data erasure.

Related tasks

Disk Erase on page 35

Wipe Confidential Data

You may have some confidential data on your hard drive in spaces where the data is stored temporarily. You may also have deleted files by using the Windows ***Recycle Bin*** and then emptying it. While you are still using your local hard drive there may be confidential information available in these unoccupied spaces.

Wiping the logical drive's deleted data does not delete existing files and folders. It processes all unoccupied drive space so that recovery of previously deleted files becomes impossible. Installed applications and existing data are not touched by this process.

When you wipe unoccupied drive space on the system disk, the process must be run under operating system booted from CD/DVD/USB disk. As a result the wipe or erase process uses an operating system that is outside the local hard drive and is not impeded by Windows system caching. This means that deleted Windows system records can be wiped clean.

Active@ KillDisk wipes unused data residue from file slack space, unused sectors and unused space in system records or directory records.

Wiping drive space can take a long time, so do this when the system is not being actively used. For example, this can be done overnight.

Related concepts

Wipe Disk concept on page 190

Related tasks

Disk Wipe on page 37

Data Recovery

Advances in data recovery have been made such that data can be reclaimed in many cases from hard drives that have been wiped and disassembled. Security agencies use advanced applications to find cybercrime related evidence. Also there are established industrial spy agencies using sophisticated channel coding

techniques such as PRML (Partial Response Maximum Likelihood), a technique used to reconstruct the data on magnetic disks. Other methods include the use of magnetic force microscopy and recovery of data based on patterns in erase bands.

Although there are very sophisticated data recovery systems available at a high price. Almost all the data can also be easily restored with an off-the-shelf data recovery utility like Active@ File Recovery, making your erased confidential data quite accessible.

Using **Active@ KillDisk** all data on your hard drive or removable device can be destroyed without the possibility of future recovery. After using **Active@ KillDisk** the process of disposal, recycling, selling or donating your storage device can be done with peace of mind.

Active@ KillDisk

Active@ KillDisk is a powerful and portable software that allows you to destroy all data on *Hard disk drive(HDD)*, *Solid-state drive (SSD)* & *USB flash drive*, excluding any possibility of deleted files and folders data recovery. **Active@ KillDisk** advanced features include:

- Enhanced visualization of physical disks and erase processes;
- Improved handling of disks with controller malfunctions;
- Stable handling of hot-swappable and dynamic disks;
- Sound notifications for completed erase jobs with different results;
- Auto hibernate or shutdown the system after all jobs are completed;
- Enhanced certificates and reports for disk erase and wipe;
- Advanced Disk Viewer with flexible Search for low-level disk inspection;
- Customizable file names for certificates & XML reports;
- Unique Computer ID can be displayed in certificates/reports;
- Disk health - *S.M.A.R.T.* information can be displayed and monitored;
- Customizable look & feel: four different application styles included;
- ATA Secure Erase option for SSD (Linux and Console packages only).

New features for version 26 include:

- Added support for the latest disk controller hardware and drivers;
- Improved compatibility with the latest Windows and Linux operating system updates;
- Added a new **Amber theme** to enhance the user experience;
- Improved **Properties Panel** for clearer and more robust data presentation;
- Expanded customization options for the Application Log/Output console;
- Scroll-wheel zooming is now supported in the Application Log/Output console and File Explorer windows;
- Updated **Disk Viewer** with advanced capabilities for accessing data storage contents;
- Various UI related tweaks and improvements;
- General stability improvements and bug fixes;
- Updated to the latest Kernel version, delivering additional performance and reliability improvements;

New features for version 25 include:

- Added erase methods IEEE Std 2883-2022 and NIST 800-88 rev.1 ;
- Minor bug fixes and functionality improvements;
- Latest kernel version including many improvements.

New features for version 24 include:

- Added Barcodes and QR codes to labels and certificates;
- Export erase results to CSV log;
- Minor bug fixes and functionality improvements;
- Latest kernel version 14.1.17 including many improvements.

New features for version 23 include:
- Improved output to certificates and application log;
- Improved submitting online requests;
- Improved work with NTFS volumes formatted with large cluster size;
- Latest kernel including bug fixes and improvements.

New features for version 15 include:
- Dark application style by default;
- Configurable dynamic disks (LDM\LVM) reconstruction;
- Device view filtering improved;
- Bug fixes and performance improvements;
- Latest kernel including bug fixes and improvements.

New features for version 14 include:
- Added context help;
- Dialogs adopted for low-resolution monitors (800x600);
- Secure e-mail notifications provided (added SSL & TLS support for SMTP);
- Improved Console functionality to support the latest hardware;
- Latest kernel including bug fixes and improvements.

New features for version 13 include:
- Resume Disk erase action to continue interrupted disk erase due to disk malfunction or errors;
- Digitally signed PDF certificate with optional encryption and visual signature presentation;
- Secure Erase (ATA command) implementation for Solid State Drives (SSD);
- Enhanced faulty disks detection and handling;
- Bug fixes and major performance improvements.

New features for version 12 include:
- Customizable Printable Labels;
- Customizable Sound Notifications;
- Redesigned and improved printable Certificates and Reports;
- Disk Serial Number can be properly detected for most scenarios, including disks connected via USB;
- Many other enhancements and stability improvements while working with unstable disks.

This release is available as an executable to run in your desktop environment, or in a bootable environment with the help of the **Active@ Boot Disk Creator** - the bootable disk creation tool included in the installation package.

Related information

Erase Methods on page 186

System Requirements

Active@ KillDisk is designed to run with the following minimum requirements:

Workstation:
- PC x64 (64-bit) compatible computer;
- Intel/AMD processor;
- 2 GB of RAM;
- 100 MB of free disk space.

Video:
- VGA (1024x768) resolution or better.

Operating Systems:

- Windows 10 or higher;
- Linux Kernel 2.x or later (all Linux family: Ubuntu, Debian etc.);

Drive Storage:

- *Compact Disc (CD)*/*DVD*/*Blu-ray* optical drive (for applicable boot disk features);
- *USB flash drive* 1.0/2.0/3.0/3.1/3.2 storage device (for applicable boot disk features);
- Disk types supported:
  - *Hard disk drive(HDD)* via *Integrated Drive Electronics (IDE)*, *AT Attachment (ATA)*, *SATA* I, SATA II, SATA III, *SAS*;
  - *Solid-state drive (SSD)* via SATA I, SATA II, SATA III, SAS;
  - *External SATA (eSATA)* and USB disks;
  - *SCSI* and *iSCSI* devices;
  - Onboard *M.2* (SATA and PCI-E types);
  - Removable media (USB drive, MemoryStick, *Secure Digital (SD) card*, Compact Flash, Zip Drive).

Active@ KillDisk supports all drives visible by the OS with read/write access, additional drivers can be loaded onto the boot disk for drivers not included by default in the bootable environment.

Related information

Installation on page 19

Software License

Active@ KillDisk is licensed **per concurrent use of the software** and **for each concurrent disk being erased or wiped** outlined in the EULA. The maximum number of disks erased in parallel corresponds to the number of purchased licenses.

One corporate license grants you an ability to run the software on one machine and erase one disk at any given time. To run on several machines in an office (or to erase multiple drives in parallel on one machine) you require the corresponding number of licenses.

Site and Enterprise licenses grant the license holder use of the software in one geographical location and worldwide respectively.

This licensing is maintained through software registration and activation. Once the commercial version of **Active@ KillDisk** is purchased the license holder will receive an email with their ***Registered Name*** and ***Registration Key***. Every machine that needs to use the fully-functional version of the software needs to be activated with this key.

Activations are limited to the number of licenses held. To transfer from one machine to another they must be deactivated from decommissioned hardware first.

For boot disks containing **Active@ KillDisk** the **Active@ Boot Disk Creator** must be registered with a registration key.

Register Online

For this task you require an active internet connection for the PC you wish to register the product on.

After installation **Active@ KillDisk** still starts as a FREE (unregistered) version having limited functionality. You need to register it first to have all professional features activated. To register the software:

1. Start registration wizard

   On application first start Registration & Licensing dialog launched by default. If freeware software has already been registered, the dialog does not appear at start. In this case click ***Registration*** menu item from ***Help*** menu.

2. Select register option

   Select the ***Register or Upgrade Software*** radio button. Read the License agreement and activate the check box to agree to the Terms and Conditions of the license. Click ***Next*** to proceed with the registration.

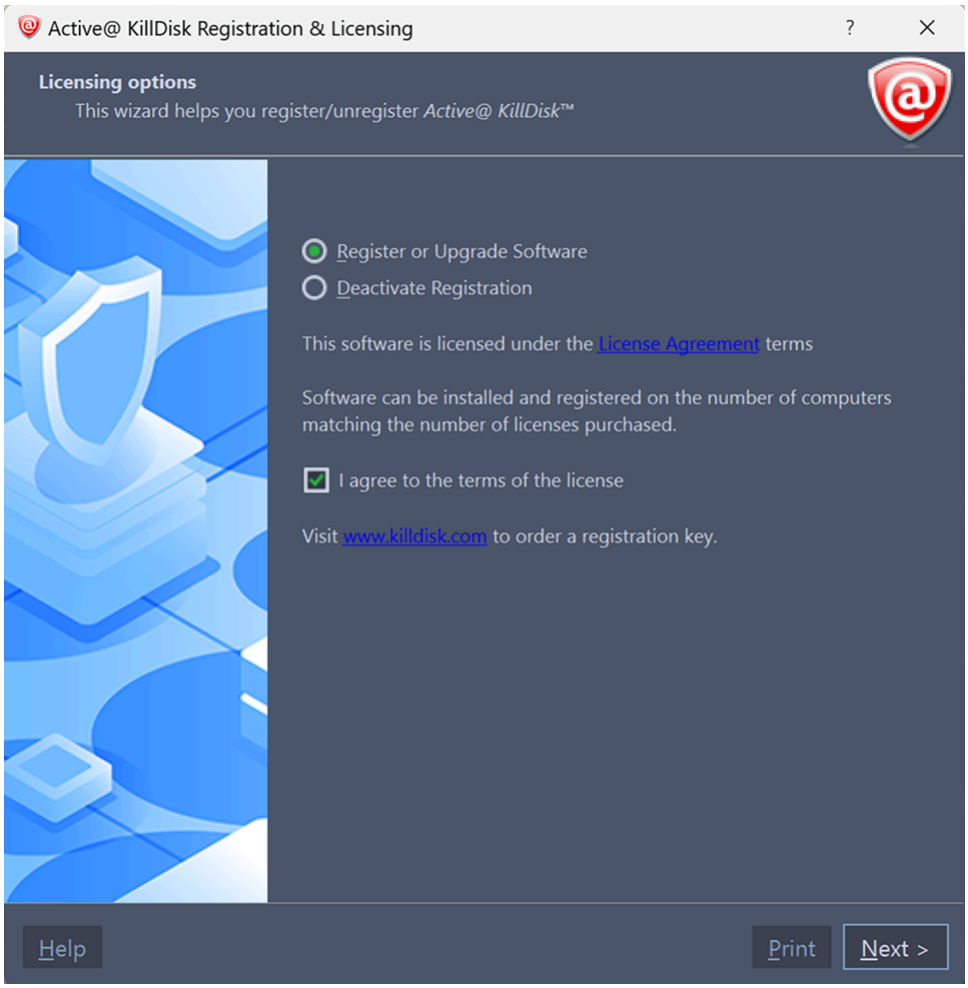


Figure 1: Registration Dialog

3. Type registration info

   Type your name into ***Registration Name*** field, then copy & paste your 30-digit registration key obtained after software purchase into the ***Registration Key*** field. Registration and network activation should be completed automatically. You should receive a response that the software has been

registered. The registration is now complete. You may click ***Next*** to go to the final confirmation and click ***Finish*** to close registration wizard.

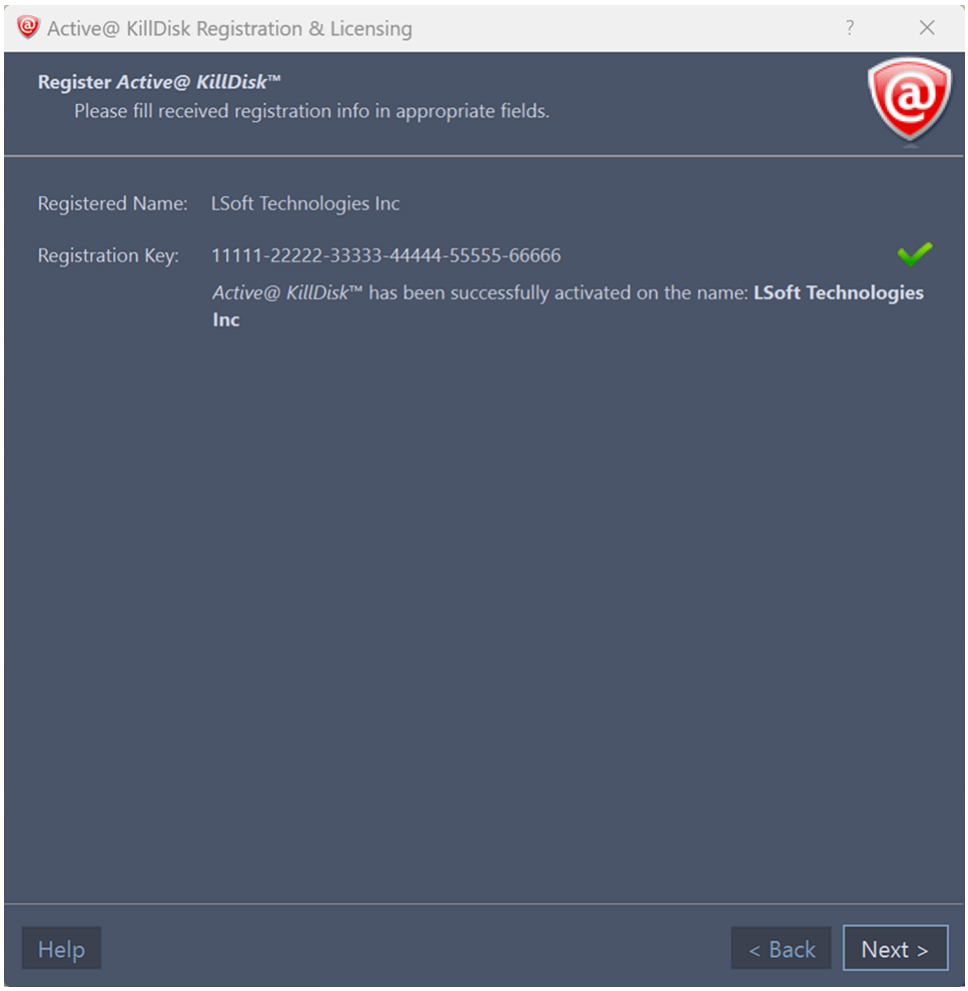


Figure 2: Registration Complete

Now you have access to all professional features of the application.

Note:

If your registration key is too long you are using the key for an earlier version. Ensure you update to the latest version by making sure your support and updates are active and use the key to this latest version. This can be done through your client profile.

Register Offline

For this method of activation, you need another PC with a web browser and active Internet connection and a USB flash disk for transferring activation data.

Note: Use this method only if the computer you are activating does not have Internet access.

In some cases such as security reasons or corporate firewalls you may not have access to an Internet connection on the machine you wish to install the software on.

For product registration and activation offline:

1. Start registration wizard

   On application first start Registration & Licensing dialog launched by default. If freeware software has already been registered, the dialog does not appear at start. In this case click ***Registration*** menu item from ***Help*** menu.

2. Select register option

   Select the ***Register or Upgrade Software*** radio button. Read the License agreement and activate the check box to agree to the Terms and Conditions of the license. Click ***Next*** to proceed with the registration.

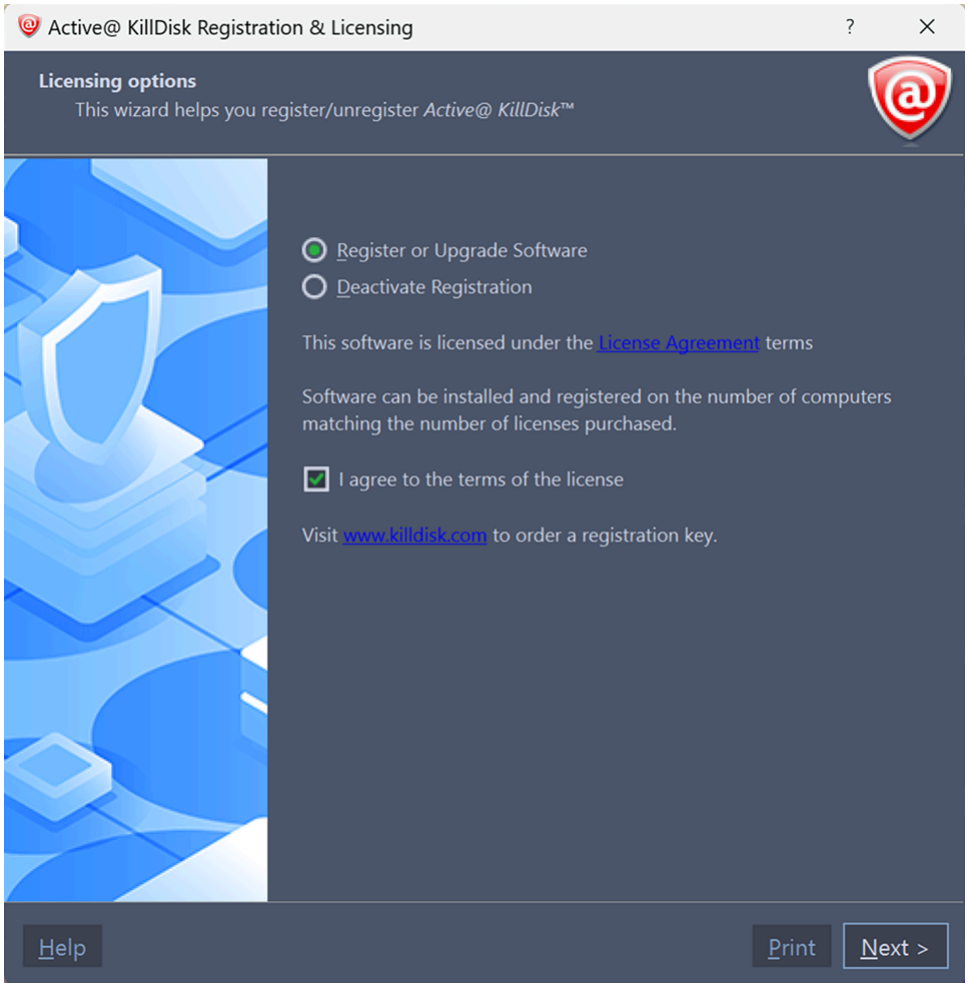


Figure 3: Registration Dialog

3. Type registration info

Type your name into ***Registration Name*** field, then copy & paste your 30-digit registration key obtained after software purchase into the ***Registration Key*** field. The Activation Request and Activation Response boxes will appear:

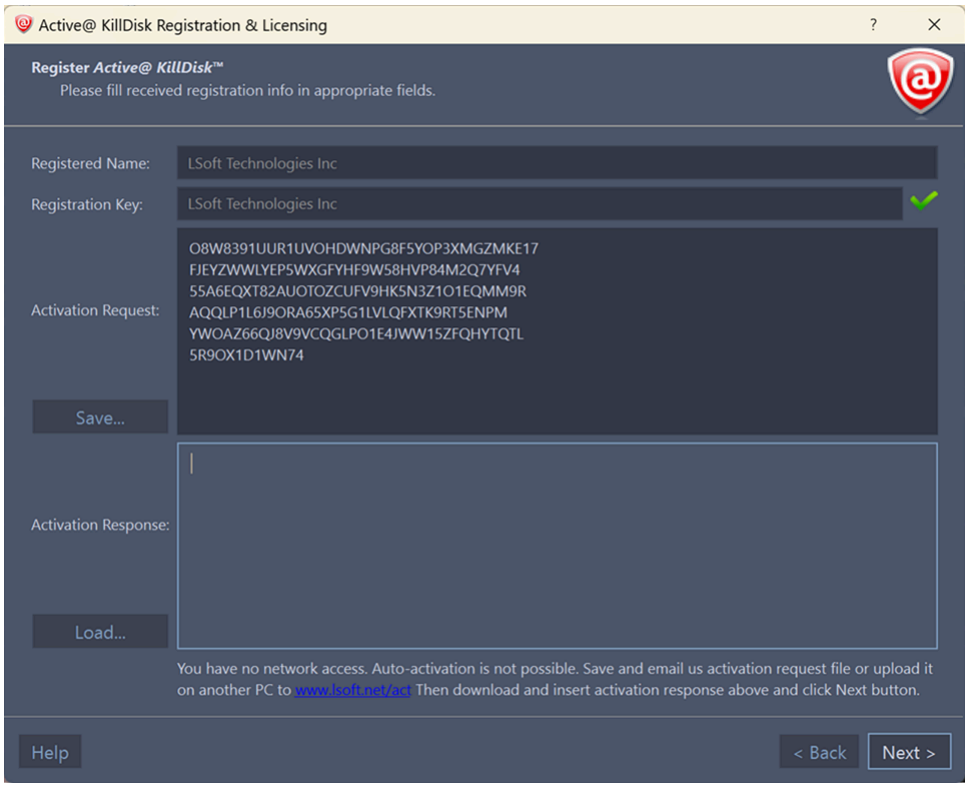


Figure 4: Offline Activation Request

Click ***Save...*** to store registration request to a file. Copy this file to a USB drive.

4. Activate via internet

   Bring the USB drive to a computer with an active Internet connection.

   Open Web browser and navigate to https://www.lsoft.net/act/

   Import the request file using the ***Choose File*** button and click ***Load***

   Click ***Process!*** to generate the Activation Response

   Save the response to your USB drive by clicking ***Save to *.licenseActivated***

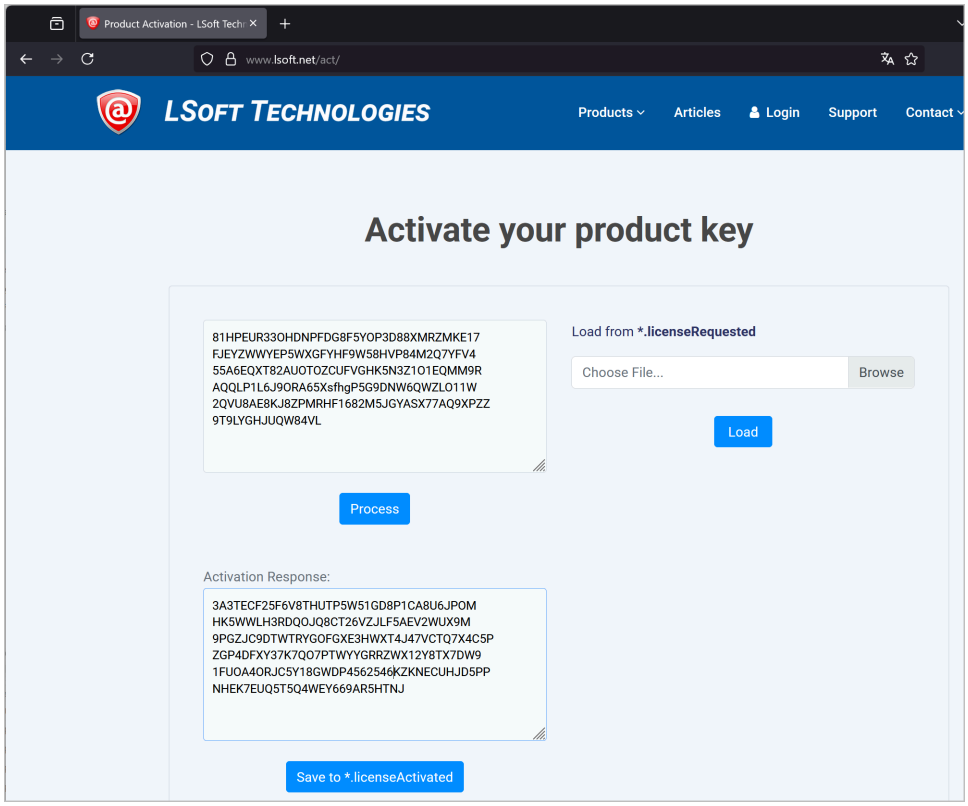


Figure 5: Activate in Browser

5. Complete activation

   Bring the USB drive to the PC with **Active@ KillDisk** installed and where activation request for offline activation was created.

   Click ***Load...*** button to import activation response from ***License File*** to the registration window and click ***Activate*** button to complete and store offline activation.

   Click ***Next*** and ***Finish*** buttons to confirm and close registration wizard.

You have now activated the software on your machine than does not have active internet connection.

Deactivate License

To transfer licenses from one machine to another you need to free up (remove) your activation on the licensed machine. You may do this by deactivating the registration from within the **Active@ KillDisk** application:

1. Start registration wizard

   Click ***Registration*** menu item from ***Help*** menu.

2. Select deactivate option

   Select the ***Deactivate Registration*** radio button. Click ***Next*** to proceed with the registration.

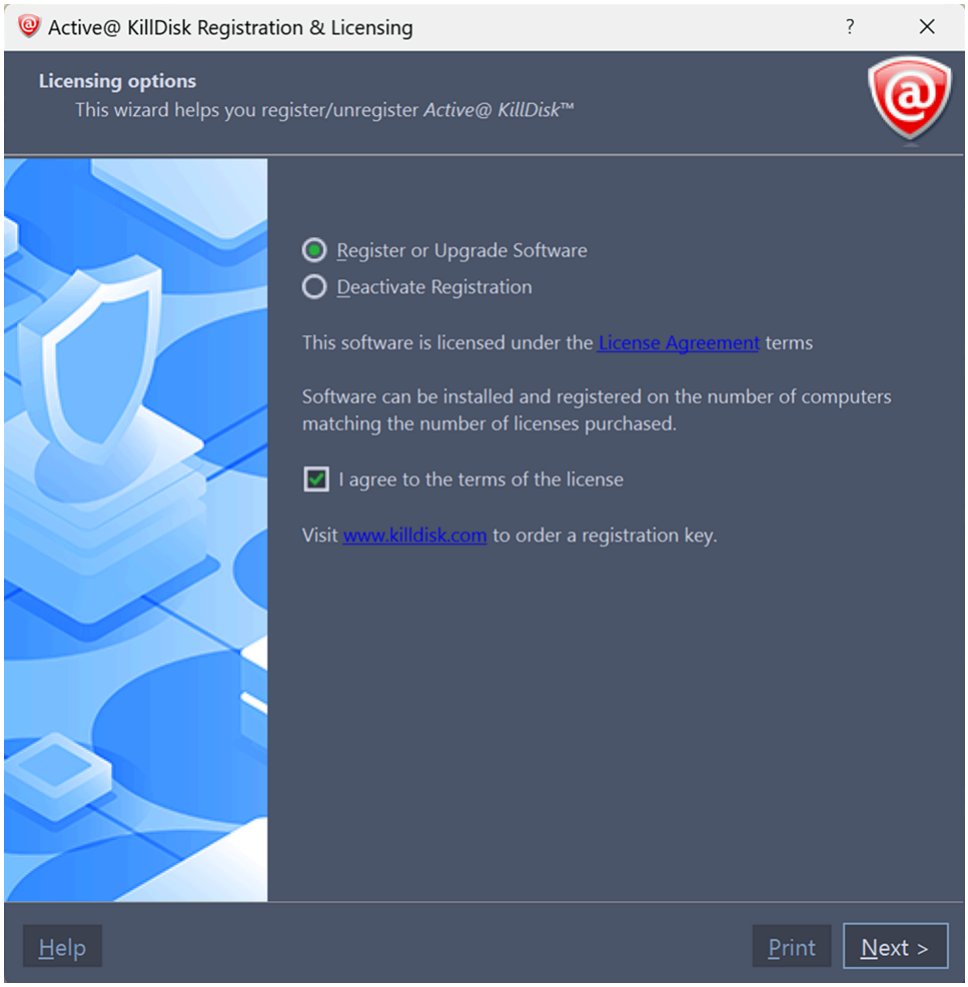


Figure 6: Registration Dialog

3. Deactivate registration

   Click ***Deactivate Registration*** button to complete online license deactivation.

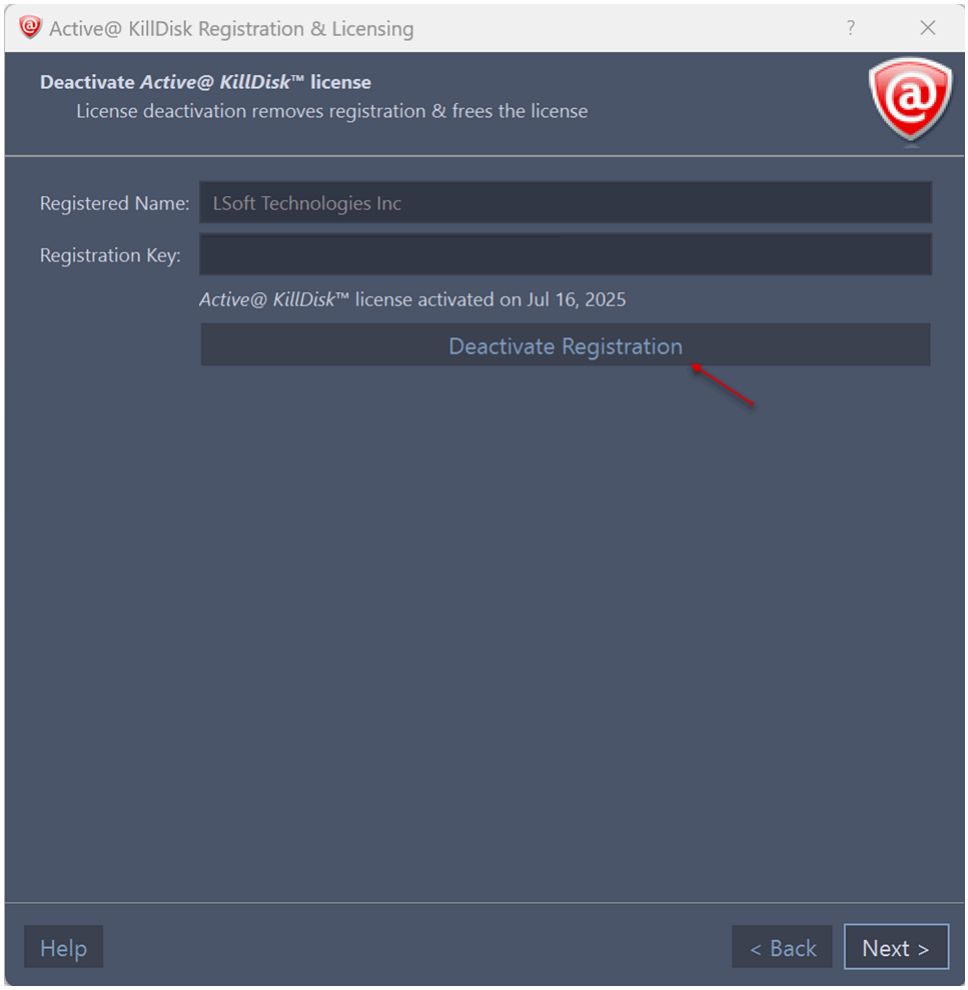


Figure 7: Deactivating Registration

Click ***Next*** and ***Finish*** buttons to confirm and close registration wizard.

Your active license is now revoked from your PC and may be used to activate a different computer.

Note:

Uninstalling the application from the computer using the uninstaller will also deactivate your license.

Installation

After purchasing **Active@ KillDisk** a registration key will be emailed to you as well as a download link to installation package named `KILLDISK-<VERSION>-SETUP.EXE` (Windows versions), `KillDiskLinux.run.tar.gz` (Linux version) or `KillDiskMacOS.dmg` (Apple macOS/OS X). This file contains everything you need to get started - just double click the file and installation wizard will take you through the setup process. You need to have Administrator's privileges to be able to install it properly.

Note:

If you purchased the Ultimate version you receive installation executable file to run on Windows. To access the Linux installation files install **Active@ KillDisk** on your Windows machine and navigate to the application directory. In `Linux` sub-folder you will find the Linux installation files. The path to the Linux application will look something like: `C:\Program Files\LSoft Technologies \Active@ KillDisk Ultimate\Linux\KillDisk_Linux_Installer.tar.gz`

After installation **Active@ KillDisk** still starts as a FREEWARE (unregistered) version having limited functionality. You need to register it first to have all professional features activated.

The installed package contains two main applications:

- **Active@ KillDisk** - Run this application to inspect local disks and erase/wipe your data;
- **Active@ Boot Disk Creator** - Create a bootable CD/DVD/BD/USB disk to boot from and run **Active@ KillDisk**. Using **Active@ KillDisk** this way allows you to wipe out confidential data from the system volumes while gaining exclusive use to partitions because the operating system runs outside the partition that you are securing.

Windows versions:

In order to install the application double click `KILLDISK-<VERSION>-SETUP.EXE` file and follow the instructions in the installation wizard.

Linux versions:

In order to install **Active@ KillDisk** make sure you found the Linux installation file as mentioned in the note above. Double click `KillDisk_Linux_Installer.tar.gz` in your Linux environment and unpack the archive to a proper location.

To start installation simply run the following command in the directory where the archive was unpacked:

```
sudo ./KillDisk_Linux_Installer.run
```

Active@ KillDisk will be installed to `/opt/lsoft/KillDisk/` and menu shortcut will be created.

Apple macOS / OS X versions:

In order to install **Active@ KillDisk** make sure you found the installation file as mentioned in the note above. Double click `KillDiskMacOS.dmg` and follow installation steps. Drag and drop **Active@ KillDisk** to `Applications` folder.

Active@ KillDisk will be installed as `/Applications/KillDisk.app` and **Active@ KillDisk** shortcut appears in `Applications` in Finder.

Note:

Boot Disk Creator and Boot Disks are not supplied for macOS/OS X because newest Apple M1/M2 processors much differ from Intel architecture and do not support Windows-based or Linux-based bootable environments. If purchased commercial version, you can still request **Boot Disk Creator for macOS /OS X** plus related Boot Disks from us, however prepared bootable media will boot and erase the only legacy Intel-based Macs (not M1/M2).

Software Updates

Active@ KillDisk has a built-in update feature to ensure you always have an access to the latest version of the application. To check for updates, use the file menu bar to navigate to ***Help*** > ***Check for Updates***

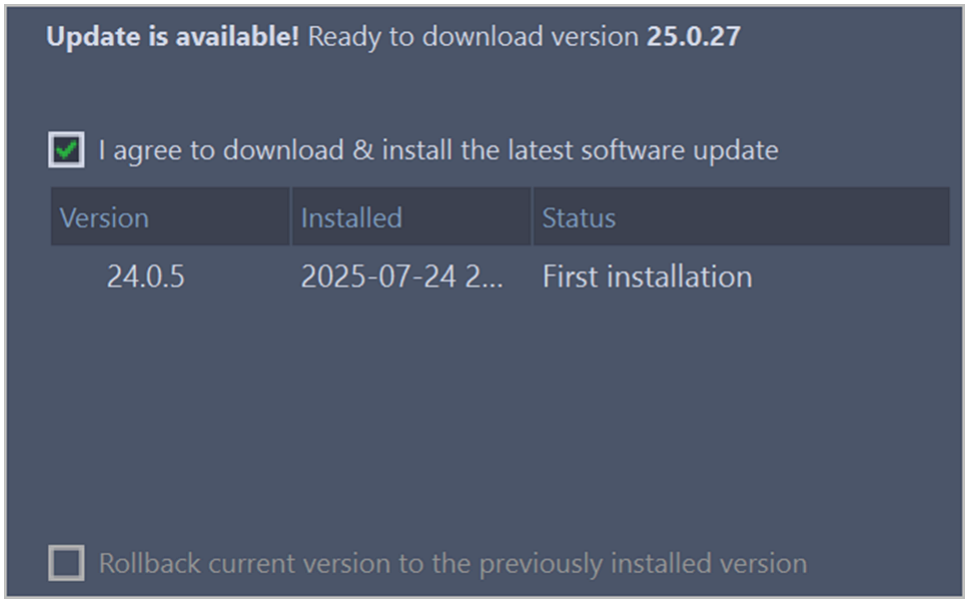


Figure 8: Check for Updates

Update dialog contains history of previously installed versions and updates.

If a new version or update is detected it can be downloaded and installed on the next wizard steps.

Click ***Next*** button to proceed with an update, if exists. Software download and installation will start automatically.

Note:

Active@ KillDisk stores your previously installed versions so you may roll back to any of your older versions at any time. To rollback to a previous version, just select the target version, mark ***the Rollback to previously installed version*** check box, and click the ***Next*** button.

Launch and Configure

When you launch **Active@ KillDisk** for the first time, the application will display the main interface with a list of all available storage devices detected on your system. Before performing any operations, it is recommended that you review the default settings to ensure they align with your requirements.

To access the application Preferences on page 72, click **Tools > Preferences** from the main menu. The Preferences dialog provides configuration options for various aspects of the application, including erasure methods, certificate templates, disk labels, and notification settings. You can adjust these settings at any time to suit your specific needs. For most users, the default settings are appropriate for standard disk management tasks.

Note:

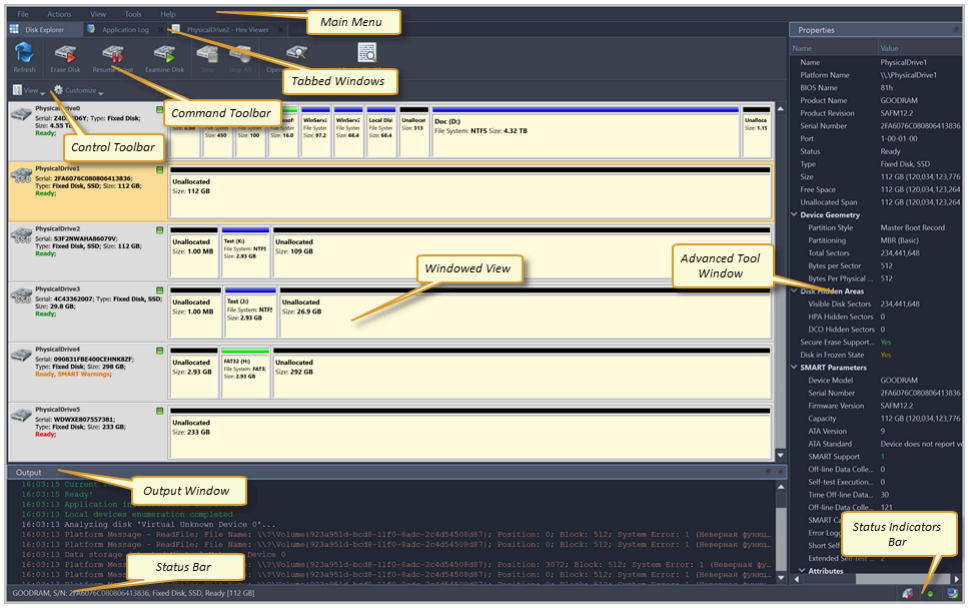
Before launching the software make sure the security USB dongle is plugged in to any USB slot.

Related information

Disk Bay Layout

Navigation

The **Active@ KillDisk** interface is organized into several key areas designed for efficient workflow and easy navigation:



Toolbar and Menu Bar

Located at the top of the window, the toolbar provides quick access to commonly used commands such as starting processes, viewing reports, and accessing preferences. The menu bar offers comprehensive access to all application features organized by category.

Tabbed Windows

Contains the window that is currently active. By default, the central area displays all detected storage devices, including internal drives, external USB drives, and network-attached storage. Each device entry shows essential information such as capacity, model name, serial number, and current status. You can select one or multiple devices to perform operations on them simultaneously.

Properties Panel

When a device is selected, the properties panel displays detailed information about the drive, including partition structure, file system type, SMART attributes, and hardware specifications. This information helps you verify that you have selected the correct device before performing any operations.

Status Bar

Located at the bottom of the window, the status bar shows the current application state, number of selected devices, network and status during active operations.

The full functionality and features of these components are discussed in corresponding sections below.

Related information

Application Log on page 24
Property View on page 116

Disk Explorer

Disk Explorer is a default workspace for the **Active@ KillDisk** application. All attached HDD/SSD/USB disks are visualized here and can be selected for different actions. Commands can be initiated from here as well as progress displayed for actions performed with disks.

This information can be presented in different views, providing different kind of access to certain features and details about the subject. Use the ***View*** drop-down menu in the command toolbar to switch between:

- Local Devices on page 23

Each view has its own customization capabilities for different use cases. Read about them on their respective pages. Use **Ctrl+Tab** or ***Ctrl+Shift+Tab*** key sequences to navigate between views.

Active@ KillDisk can automatically detect plugged removable devices and shows them in Disk Explorer. However, if plugged device does not appear in click ***Refresh*** button in toolbar to update Disk Explorer view or press and hold ***Ctrl*** button on keyboard and click ***Refresh*** button in toolbar to completely rescan and refresh all connected local data storage's.

Select disk partition volume or other object in Disk Explorer and use available command from views toolbar or ***Action*** main menu to perform an action.

Additional tool views such as Output view or Property View available through ***View*** > ***Windows*** main menu.

Related information

Preferences on page 72
Output View on page 25
Property View on page 116

Local Devices

Local Devices view shows all disks recognized by the Operating System as a flat list. All attached HDD/SSD/USB disks are visualized here and can be selected for different actions. Majority of commands can be initiated from here as well as progress displayed for actions performed with disks.

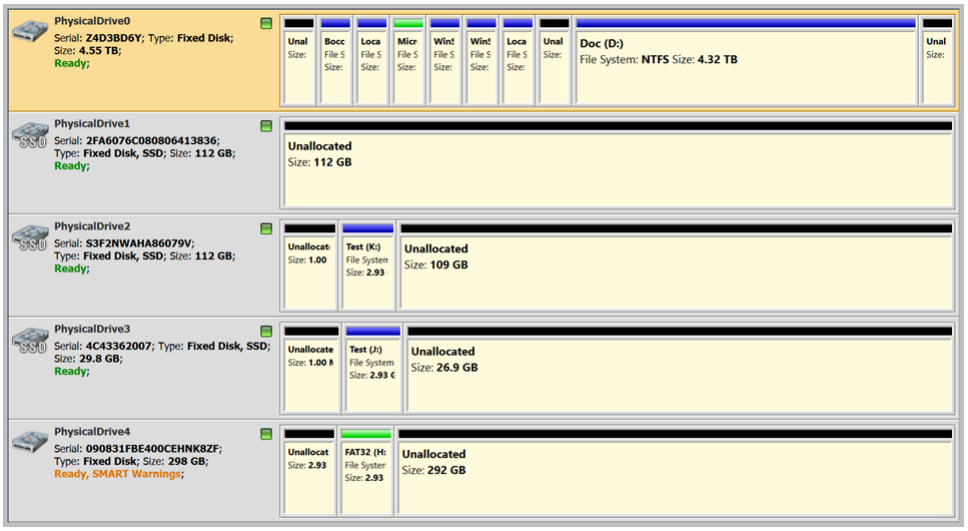


Figure 9: Local Devices View

Select disk partition volume or other object in Disk Explorer and use available command from views toolbar or ***Action*** main menu to perform an action.

Device view mode has some customization options to adjust:

Customize menu

Show System Devices

Displays the disk where Operating System is installed. This is off by default to prevent accidental erasure of the system

Show Not Ready Devices

Displays devices not yet initialized and used by Operating System

Show Removable Devices

Displays all removable and externally connected disks (such as USB Flash Drives and External USB Disks)

Compact View

Changes the layout of the Disk View from display block to inline block orientation

Related information

Preferences on page 72
Output View on page 25
Property View on page 116

Application Log

Application Log reflects every action taken by the application and displays messages, notifications and other service information. Use these messages to observe and analyze processes flow.

To open Application log view do one of the following:

- Click ***Tools*** > ***Application Log*** from the main menu
- Press ***F8*** keyboard shortcut

Once Application Log View is open and active, you can use toolbar buttons and the context menu to perform the following tasks:

Save log as

Opens a standard ***Save As*** dialog. Save the actual application log file to the local disk Default is .LOG file extension.

Save hardware info as

Opens a standard ***Save As*** dialog. Save the disk diagnostic file to the local disk. Default is .XML file extension.

Log entry filter

Shows or hides specific entry types in Log View, such as System, Network or Security.

Text size

Changes text size from Small to Large. You can also use press and hold ***CTRL*** key and mouse wheel to adjust text to desired size.

Expand/Collapse All

Expands all collapsed log nodes.

Clear

Clear the log for the current application session.

Tip:

We recommend that you attach a copy of the log file to all requests made to our technical support group. The entries in this file will help us to resolve certain issues.

Output View

Simplified log view pane can be used as a convenient way to view application events of ongoing processes at any featured view. Context menu allows to filter out event entries by its type or significance (priority) attributes. This pane remains visible when switching between main featured views.

To open Output View do one of the following:

- Click ***View*** > ***Windows*** > ***Output*** from the main menu or
- Press ***CTRL + O*** keyboard shortcut

Related information

Disk Explorer on page 23
Property View on page 116

Create a Boot Disk

Boot Disk Creator helps you to prepare a bootable CD/DVD/BD or USB storage device that you may use to boot any PC and use **Active@ KillDisk** to destroy all data on the attached HDD and SSD.

To prepare a bootable disk:

1. Run Boot Disk Creator

   Run application from the Windows Start menu on Windows platform. **Boot Disk Creator** wizard appears.

2. Register software

   This is an optional step: if software hasn't been registered yet, you need to register software first. Click ***Registration*** link in the bottom-left corner.

3. Select bootable media

   Select desired bootable media to be created: ***CD/DVD/Blu-ray Disc***, ***USB Flash Drive*** or ***ISO Image File***. If several media disks are inserted, open Flash Drives combo box and choose a particular device. Click ***Next*** to proceed to the next step.

   Note:

   A USB Drive or blank CD/DVD/BD must be inserted and explicitly chosen on the first step before you can proceed further.

   Note:

   If inserted USB Flash Drive doesn't appear in drop-down list, click ***USB not listed: Initialize Disk*** link. You should be able to find removable disk in list of all attached devices and initialize it to make compatible with the application. Initialization process **destroys all data** on the selected USB device.

4. Select target platform

   Select the target platform for booting up: ***Windows-based Boot Disk***, ***Linux-based LiveCD/LiveUSB*** or ***Console-based Boot Disk*** (text mode application). Depending on version purchased one or more target platforms are available for selection. Click ***Next*** to proceed to the next step.

5. Configure boot disk

   Specify additional and customized boot disk options:

   a) Configure System Boot Settings

      To customize boot options click the ***System Boot Settings*** tab. You can change the default settings to be used: ***Time Zone***, ***Additional Language***, ***Display Resolution***, ***Display Scale***, ***Default Application Start*** and ***Auto-start Delay***.

      Network and ***Security*** sub-tabs allow to configure IP & Firewall settings, mapping a network resource and protecting Boot Disk with a password.

   b) Add your files

      To add your custom files to the bootable media click ***User's Files*** tab. Add files or folders with files using the related buttons at the right side. Added items will be placed in the ***User_Files*** folder at the root of bootable media.

   c) Add extra drivers

      To add specific drivers to be loaded automatically, click ***Add Drivers*** tab. Add all files for the particular driver (*.INF, *.SYS, ...). Added items will be placed in the ***BootDisk_Drivers*** folder at the root of bootable media. At boot time all *.INF files located in this folder will be installed (if compatible with a platform).

   d) Add scripts

      To add specific scripts to be launched after Boot Disk is loaded, click ***Add Scripts*** tab. Add your scripts (*.CMD files). Added files will be placed in the ***BootDisk_Scripts*** folder at the root of

bootable media. At boot time all *.CMD files located in this folder will be executed (if properly created for the particular platform).

e) Add command line parameters

To add command line parameters for **Active@ KillDisk** start up, click ***Application Startup*** tab and type all parameters required (read documentation first). This tab is only enabled when **Active@ KillDisk** has been set up as a ***Default Application*** at Boot Disk start up.

f) Configure KillDisk environment

To specify paths for **Active@ KillDisk** Certificate Logo file (LPG/PNG/BMP), Settings file (XML), Digital Signature file (PFX) and Volume name to store Certificates/Logs/Reports, click ***App Config*** tab and enter configuration information. You can configure storing Erase Certificates and reports to a mapped network share.

Click ***Next*** to proceed to the final step.

6. Create Boot Disk

Verify selected media, boot up environment and **Active@ KillDisk** configuration. Click ***Create*** button to start bootable media creation process. A progress bar appears while the media is being prepared. Click ***Finish*** to exit the wizard.

You can use prepared bootable media to boot up any 64-bit PC and erase/wipe all attached disks.

Note:

Not all the additional boot disk options are accessible for all platforms. For example, *Add Drivers* section is available to Windows Operating System only.

Related information

Software License on page 12
Common Tips on page 126
Command Line Mode on page 47

KillDisk and PXE

Active@ KillDisk software can be loaded over the network via a PXE environment. To do this, the **Active@ KillDisk** software and settings needs to be added to the image for network loading and the network environment must be correctly configured. The main points are described in this section.

Related tasks

Place Active@ KillDisk into a WinPE image on page 27
Load Active@ KillDisk over the network via PXE environment on Windows Server platform on page 29
Load Active@ KillDisk over the network via PXE environment on a Windows 10 computer on page 30
Configuring DHCP and TFTP settings on page 31
Creating a BCD file on page 33

Place Active@ KillDisk into a WinPE image

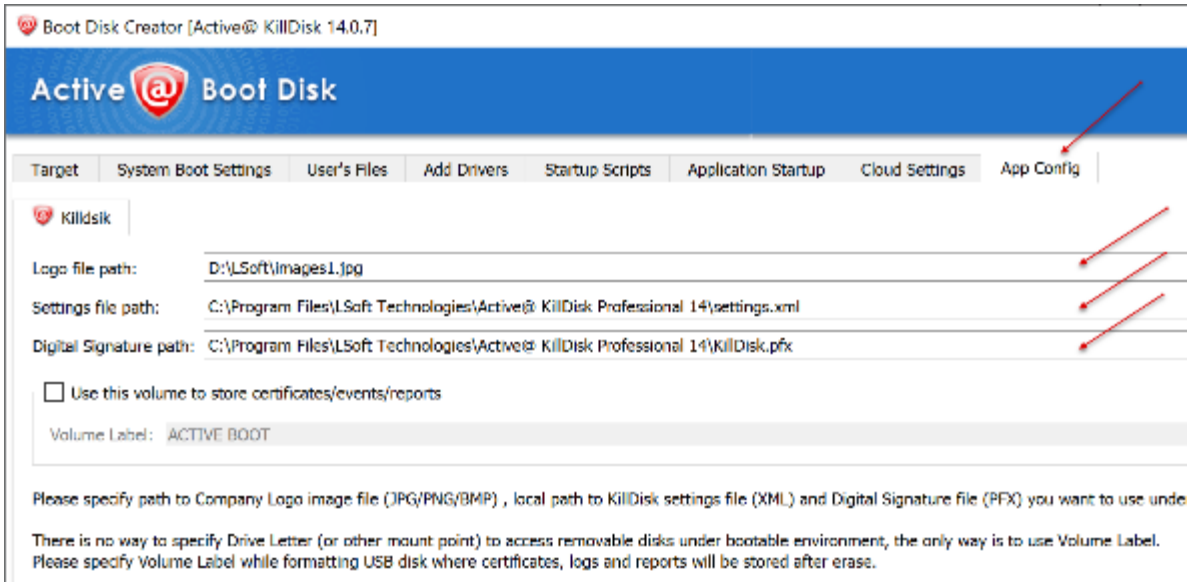
For place a registered **Active@ KillDisk** into a WinPE image for use in a network PXE boot environment need to do next steps:

Note:

To modify WinPE image (WIM) you need to have Windows ADK installed.

1. Create a bootable media.

   Start the **Boot Disk Creator** form Windows Start menu and prepare a bootable media. For KillDisk settings fill in the data on App Config page.



   Let's assume that the **Boot Disk** media has an F: letter in our environment.

2. Run **Command Prompt** as an Administrator.
3. Mount `BOOT.WIM` file.

   Create an empty directory C:\MOUNT and mount `BOOT.WIM` file using the DISM tool command:

```
Dism /mount-image /imagefile:F:\sources\boot.wim /index:1 /mountdir:C:\mount
```

4. Replace `BOOTDISK.KEY`.

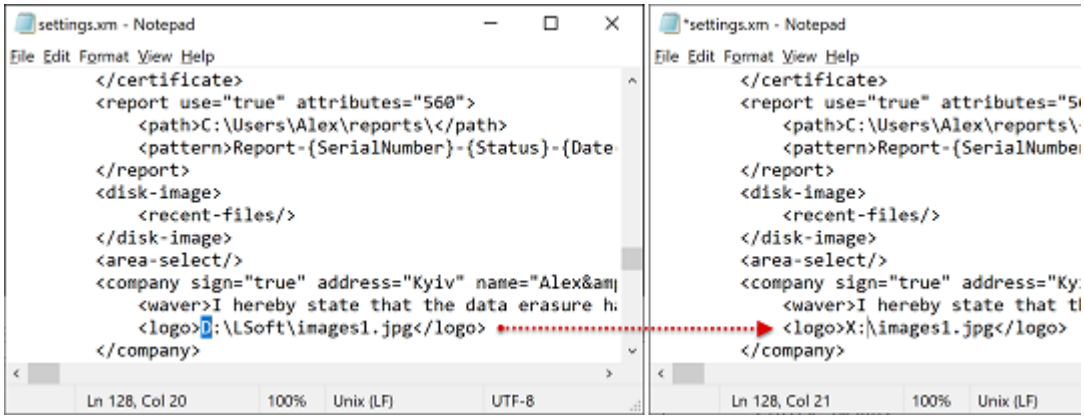
   Replace `BOOTDISK.KEY` in `C:\MOUNT` directory with `BOOTDISK.KEY` located at the root of **Boot Disk** media (`F:\ BOOTDISK.KEY`). This file is required and contains user's registration information.

5. Copy logo

   Copy your company logo file from Boot Disk media (located `F:\_kd\images1.jpg`) to `C:\MOUNT` directory.

6. Edit `settings.xml`

   Open `settings.xml` file on Boot Disk media (`F:\_kd\settings.xml`) for edit using Notepad. Change path for your company logo file to `X:\`.



   Copy modified `settings.xml` to `C:\MOUNT\PROGRAM FILES\BOOTDISK\`.

7. Create the folder with scripts if needed

   Create folder `BootDisk_Scripts` in `C:\MOUNT\PROGRAM FILES\BOOTDISK\`. In any text editor (Notepad) create script file and copy it to the newly created folder `BootDisk_Scripts`. For example:

```
cd x:\program files\bootdisk\
call killdisk.exe -em=0 -p=1 -v=11 -eh=5 -id -fp -bp
```

8. Dismount the `BOOT.WIM`

   Dismount the `BOOT.WIM` image and commit the changes you applied:

```
Dism /Unmount-Image /MountDir:C:\mount /commit
```

Use prepared F:\SOURCES\BOOT.WIM as a network PXE boot environment.

Related tasks

Load Active@ KillDisk over the network via PXE environment on Windows Server platform on page 29
Load Active@ KillDisk over the network via PXE environment on a Windows 10 computer on page 30
Configuring DHCP and TFTP settings on page 31
Creating a BCD file on page 33

Load Active@ KillDisk over the network via PXE environment on Windows Server platform

For load **Active@ KillDisk** over the network via PXE environment on Windows Server platform proceed as follows:

1. Add roles Windows Deployment Services.
2. Configure the WDS server, but don't add images in WDS Configuration Wizard.
3. Add Windows PE image with **Active@ KillDisk** software Boot.wim in Boot Images on WDS server.
4. In properties of WDS server in Boot tab add our image as default boot image for x64 architecture.
5. Configure the DHCP server for work with WDS server.

For more detailed instructions, read Microsoft TechNet official documentation.

Related tasks

Place Active@ KillDisk into a WinPE image on page 27
Load Active@ KillDisk over the network via PXE environment on a Windows 10 computer on page 30
Configuring DHCP and TFTP settings on page 31
Creating a BCD file on page 33

Load Active@ KillDisk over the network via PXE environment on a Windows 10 computer

There are several steps required to do this: configuring the WinPE WIM, Boot Manager and PXE Server.

For the configuration steps, let's assume that inserted **Boot Disk** has a `F:` letter in our configuration environment.

1. Copy WinPE Source Files onto the PXE Server
   a. Map a network connection to the root TFTP directory on the PXE/TFTP server and create a `\BOOT` folder there. We will assign this network drive the `Y:` letter.

      Note:

      You can the **'Easy access'** feature in the **Windows Explorer** to do this. Make sure to enable read/write permissions in the sharing and folder options.

   b. Copy the PXE boot files from the mounted `\BOOT` folder of the **Active@ Boot Disk** `boot.wim` to the `\BOOT` folder on PXE/TFTP server. The command for example:

      ```
      copy C:\mount\windows\boot\pxe\*.* y:\boot
      ```

      Note:

      To mount/dismount the `boot.wim` file, see Place Active@ KillDisk into a WinPE image on page 27.

   c. After dismounting the `boot.wim`, copy the bootable Windows PE image (`F:\ Sources \boot.wim`) to the `\BOOT` folder on PXE/TFTP server.
   d. Copy the file `boot.sdi` (`F:\Boot\boot.sdi`) to the `\BOOT` folder on PXE/TFTP server.
2. Configure boot configuration
   a. On a Windows 10 computer or in a Windows PE environment, create a BCD store using the **BCDEdit** tool.
   b. In the BCD store, configure the RAMDISK, BOOTMGR and OS Loader settings for the Windows PE image.
   c. Copy the BCD file to the `\BOOT` folder on PXE/TFTP server
   d. Configure your PXE/TFTP server and DHCP server to point PXE clients to download `PXEBoot.com` or `PXEBoot.n12`.

These are a few of the files that were copied over to the server in Step 1.

For more details, see Creating a BCD file on page 33.

3. Deployment process

   Boot the client machine through PXE, connected to the network. After pressing initializing the PXE boot, the system should handle the rest. Here's what will happen:

   a. The client is directed (by using DHCP Options or the PXE Server response) to download `PXEBoot.com`.
   b. `PXEBoot.com` downloads `Bootmgr.exe` and the BCD store. The BCD store must reside in a `\BOOT` directory in the TFTP root folder. Additionally, the BCD store must be called BCD.
   c. `Bootmgr.exe` reads the BCD operating system entries and downloads `boot.sdi` and the Windows PE image.
   d. `Bootmgr.exe` begins booting Windows PE by running Winload.exe within the Windows PE image.

For more detailed instructions, read the Microsoft TechNet official documentation.

Related tasks

Place Active@ KillDisk into a WinPE image on page 27
Load Active@ KillDisk over the network via PXE environment on Windows Server platform on page 29
Configuring DHCP and TFTP settings on page 31
Creating a BCD file on page 33

Configuring DHCP and TFTP settings

Configuring a TFTP server is made simple with a tool called **Serva**. You can download it here.

This tool is an "Automated PXE Server Solution Accelerator" that supports a variety of server protocols. The ones we will be configuring are TFTP and DHCP.

1. Click the logo in the top left to access the Settings.

2. Configure DHCP settings

Configure your DHCP settings. You may copy the ones below, just make sure the address it binds to is a static IP address from your router. Under IP Pool 1st addr, input the first available IP address in your routers IP pool settings.

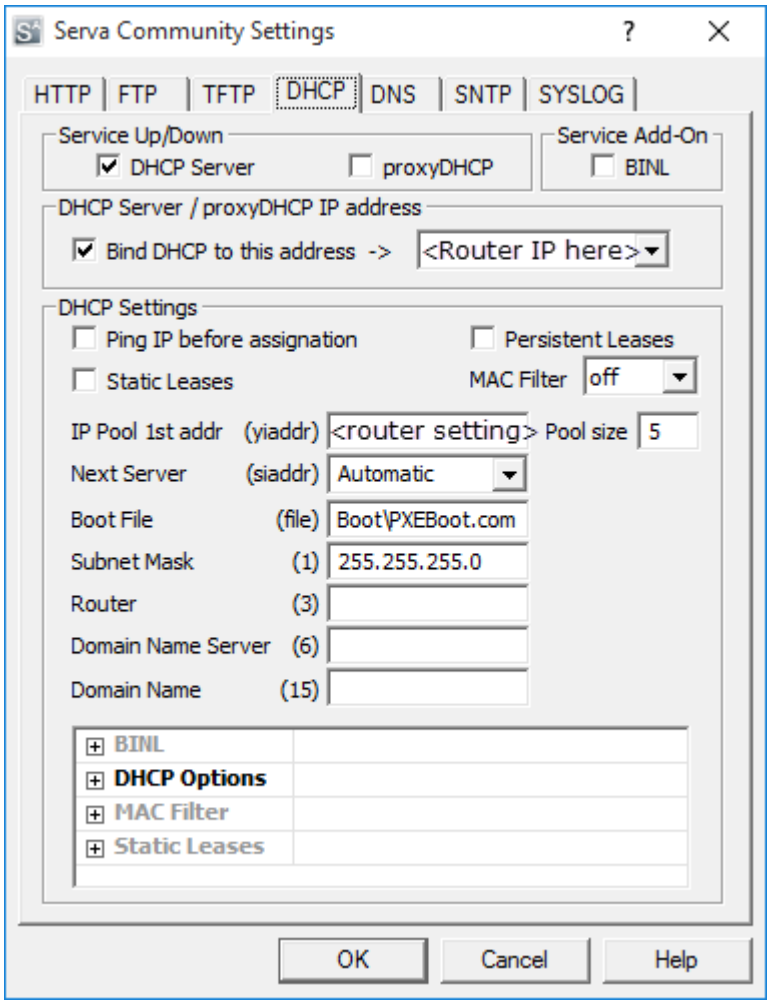


Figure 10: DHCP Configuration

3. Configure TFTP settings.

   Configure your TFTP settings. You may also copy the setting below. Again, make sure the IP address is your router's static IP and the TFTP server root directory is the one you configured in Step 1.

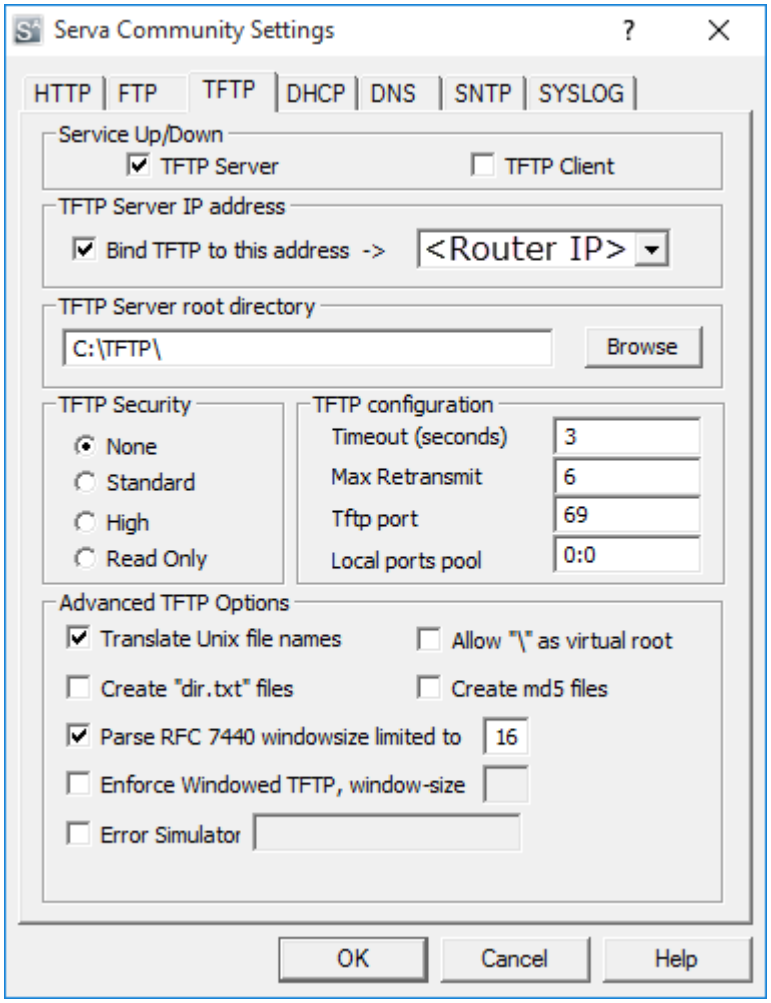


Figure 11: TFTP Configuration

Once the settings are configured, reset the application and your PXE server should be fully operational!

Related tasks

Place Active@ KillDisk into a WinPE image on page 27
Load Active@ KillDisk over the network via PXE environment on Windows Server platform on page 29
Load Active@ KillDisk over the network via PXE environment on a Windows 10 computer on page 30
Creating a BCD file on page 33

Creating a BCD file

This entire process is done in **Command Prompt**. Be sure to run it as Administrator.

1. Create a BCD store using **bcdedit.exe**.

```
bcdedit /createstore c:\BCD
```

2. Configure RAMDISK settings.

```
bcdedit /store c:\BCD /create {ramdiskoptions} /d "Ramdisk options"
bcdedit /store c:\BCD /set {ramdiskoptions} ramdisksdidevice boot
bcdedit /store c:\BCD /set {ramdiskoptions} ramdisksdipath \boot\boot.sdi
bcdedit /store c:\BCD /create /d "winpe boot image" /application osloader
```

The last command will return a GUID. For example:

```
The entry { bb254249-93e9-11e7-84cb-6c71d9da760e } was successfully created.
```

Copy this GUID for use in the next set of commands. In each command shown, replace "GUID1" with your GUID.

3. Create a new boot application entry for the Windows PE image.

```
bcdedit /store c:\BCD /set {bb254249-93e9-11e7-84cb-6c71d9da760e} device ramdisk=[boot]\Boot\boot.wim,
{ramdiskoptions}
bcdedit /store c:\BCD /set {bb254249-93e9-11e7-84cb-6c71d9da760e} path \windows\system32\winload.exe
bcdedit /store c:\BCD /set {bb254249-93e9-11e7-84cb-6c71d9da760e} osdevice ramdisk=[boot]\Boot\boot.wim,
{ramdiskoptions}
bcdedit /store c:\BCD /set {bb254249-93e9-11e7-84cb-6c71d9da760e} systemroot \windows
bcdedit /store c:\BCD /set {bb254249-93e9-11e7-84cb-6c71d9da760e} detecthal Yes
bcdedit /store c:\BCD /set {bb254249-93e9-11e7-84cb-6c71d9da760e} winpe Yes
```

4. Configure BOOTMGR settings (remember to replace GUID1 in the third command with your GUID).

```
bcdedit /store c:\BCD /create {bootmgr} /d "boot manager"
bcdedit /store c:\BCD /set {bootmgr} timeout 30
bcdedit /store c:\BCD -displayorder {bb254249-93e9-11e7-84cb-6c71d9da760e} -addlast
```

5. Copy the BCD file to your TFTP server.

```
copy c:\BCD \\PXE-1\TFTP\Boot\BCD
```

Your PXE/TFTP server is now configured. You can view the BCD settings that have been configured using the command:

```
bcdedit /store <BCD file location> /enum all
```

Example of BCD settings:

```
C:\>bcdedit /store C:\BCD /enum all
Windows Boot Manager
--------------------
identifier              {bootmgr}
description             boot manager
displayorder            {bb254249-93e9-11e7-84cb-6c71d9da760e}
timeout                 30

Windows Boot Loader
-------------------
identifier              {bb254249-93e9-11e7-84cb-6c71d9da760e}
device                  ramdisk=[boot]\boot\boot.wim,{ramdiskoptions}
description             winpe boot image
osdevice                ramdisk=[boot]\boot\boot.wim,{ramdiskoptions}
systemroot              \Windows
detecthal               Yes
winpe                   Yes

Setup Ramdisk Options
---------------------
identifier              {ramdiskoptions}
description             ramdisk options
ramdisksdidevice        boot
ramdisksdipath          \boot\boot.sdi
```

Note:

Your GUID will be different than the one shown above.

Related tasks

Place Active@ KillDisk into a WinPE image on page 27
Load Active@ KillDisk over the network via PXE environment on Windows Server platform on page 29
Load Active@ KillDisk over the network via PXE environment on a Windows 10 computer on page 30
Configuring DHCP and TFTP settings on page 31

Using Active@ KillDisk

To perform a typical disk operation in **Active@ KillDisk**, follow this general workflow: select the target device from the Local Devices on page 23 list, choose the desired operation from the toolbar or menu, configure the operation settings in the dialog that appears, review your selections carefully, and click **Start** to begin the process. Throughout the operation, progress information will be displayed in the status bar and device list, allowing you to monitor the task in real time.

Usage scenarios include: Disk Erase, Disk Wipe, Secure Erase, Certificates, Labels, Reports, Command Line Mode, and Batch Mode operations.

Refer to the Application Log on page 24 for ongoing process details and results. The Processing Summary on page 46 dialog will provide complete summary information and the ability to print it in different formats.

Disk Erase

Individual disks or disks partitions can be erased with just few clicks using many international sanitizing standards.

1. Commence task

   In Disk Explorer on page 23 open Local Devices on page 23 view to select one or more physical disks or partitions to erase. Use ***Ctrl+Left Mouse Click*** for multiple selection. To select all attached disks, press ***Ctrl+A***.

   Initiate the *Disk Erase* task using one of the following methods:

   - Click the ***Erase Disk*** command on the action toolbar;
   - Select ***Actions*** > ***Erase Disk*** from the main menu;
   - Click ***Erase Disk*** from the context menu.

2. Configure task options

   If necessary, configure task options on each page of the Erase Disks dialog, which includes the following:

   - Disk Erase on page 74
   - Erase Certificate on page 77
   - Processing Report on page 80
   - Processing CSV Log on page 81
   - Error Handling on page 85

   If a single disk is selected to perform the task, the exact area for erasure can be optionally specified on the Disk Area Selection on page 36 page. Otherwise, use the Selected Disks on page 87 page to adjust disk or partition selection.

3. Execute task

   Click the ***Start*** button to proceed to the final Confirm Critical Actions on page 75 dialog. This is an additional precautionary measure. If you proceed with confirmation, all data on the selected disk(s) or selected disk area will be destroyed permanently without any possibility of recovery.

   Note:

   Depending on settings, this dialog can be skipped.

   After starting the erase operation, a progress bar is displayed in the disk area. The progress bar represents the percentage of disk space being sanitized. As the procedure progresses, the percentage increases and the time remaining is recalculated.

   You can terminate processing for all disks at once by clicking the ***Stop All*** button (via the action toolbar, main menu, or context menu) or for individual disks by selecting the disk and clicking ***Stop***.

After the process finishes, each disk will be highlighted with the overall result (**Success**, **Failed**, **Canceled**, etc.) displayed at the top of the disk in different colors.

By default, Processing Summary on page 46 dialog will appear with detailed information of erasure results for each disk, including complete task attributes, log etc.

Note:

Depending on settings, this dialog can be skipped.

Related information

Erase Methods on page 186
User Defined erase method
Processing Summary on page 46
Generated Documents on page 52

Disk Area Selection

Active@ KillDisk has an option to specify a particular area on the disk to erase. To access this feature you have to select the single disk first. In ***Local Devices*** view initiate the ***Erase Disk*** command.

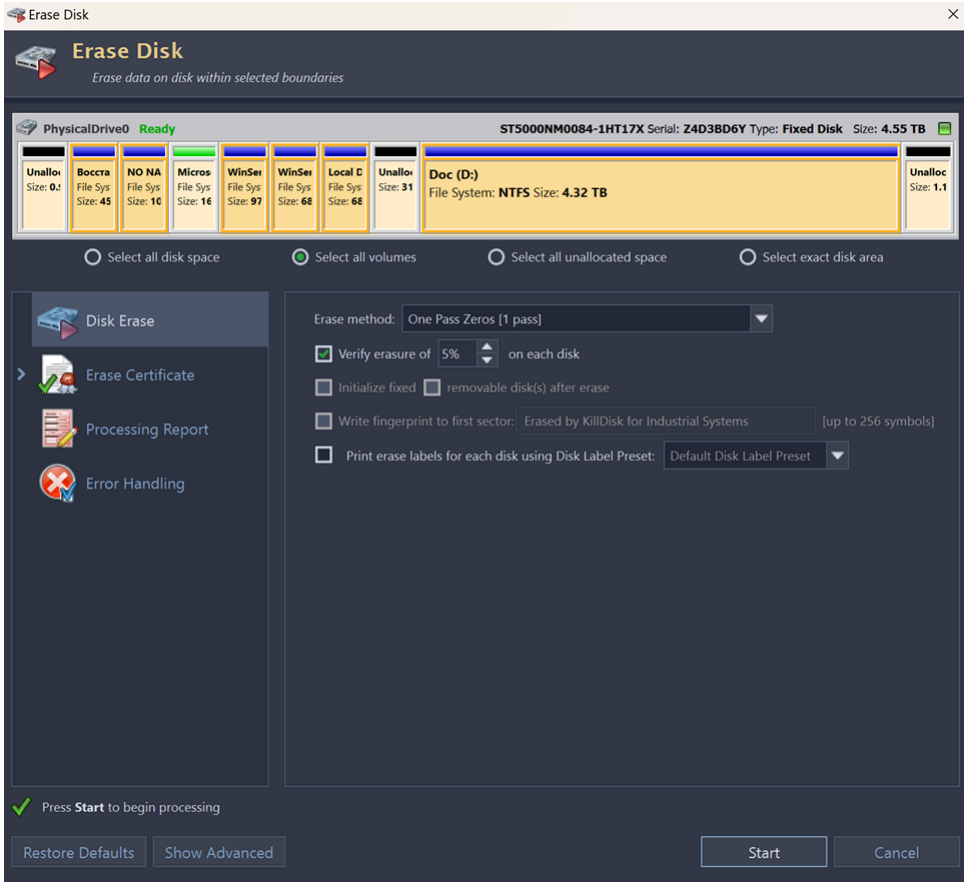


Figure 12: Erase a Specific Area

Disk area options for the erase are:

Select all disk space
This is a default disk area option and applies to entire disk surface to be erased;

Select all volumes
This option erases all existing volumes and partitions on the disk;

Select all unallocated space
This option erases only unallocated disk areas, where volumes and partitions do not exist;

Select exact disk area
Use sliders on the disk visualization in order to select a particular range of sectors. You can also click on sector numbers and type particular sectors manually.

You may also click on individual partitions and they will be selected for erasure.

Disk Wipe

When you select a physical device, the ***Wipe*** command processes all logical drives consecutively, erasing data in unoccupied areas (free clusters and system areas) while leaving existing data intact. Unallocated space where no partitions exist is also erased.

Note:

If you want to erase ALL data (both existing and deleted files) from the device permanently, use Disk Erase on page 35.

If the **Active@ KillDisk** d application detects that a partition has been damaged, it does not wipe data in that area because the partition might contain important data. In some cases, partitions on a device cannot be wiped. Examples include an unknown or unsupported file system, a system volume, or an application startup disk. In these cases, the ***Wipe*** command is disabled. If you select a device and the ***Wipe*** button is disabled, select individual partitions (volumes) and wipe them separately.

Disk Wipe complete process is described below.

1. Commence task

   In In Disk Explorer on page 23 open Local Devices on page 23 view to select one or more disk partitions to wipe. Use ***Ctrl+Left Mouse Click*** for multiple selection.

   Initiate the Wipe task using one of the following methods:

   - Click the ***Wipe Disk*** command on the action toolbar;
   - Select ***Actions*** > ***Wipe Disk*** from the main menu;
   - Click ***Wipe Disk*** from the context menu.

2. Configure task options

   If necessary, configure task options on each page of the Erase Disks dialog, which includes the following:

   - Disk Wipe on page 76
   - Erase Certificate on page 77
   - Processing Report on page 80
   - Processing CSV Log on page 81
   - Error Handling on page 85

   If a single disk is selected to perform the task, the exact area for wipe can be optionally specified on the Disk Area Selection on page 36 page. Otherwise, use the Selected Disks on page 87 page to adjust disk or partition selection.

3. Execute task

   Click ***Start*** button to reach the final step before wiping out deleted data. Click ***Yes*** to confirm ***Wipe*** action and process starts.

   After starting the erase operation, a progress bar is displayed in the disk area. The progress bar represents the percentage of disk space being wiped. As the procedure progresses, the percentage increases and the time remaining is recalculated.

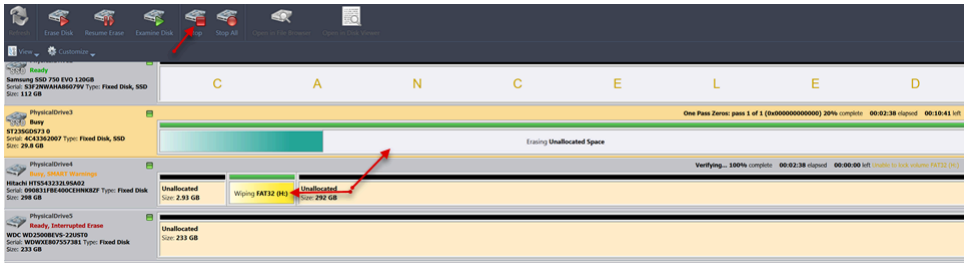


Figure 13: Disk Wipe Progress

   The progress of the wiping procedure is displayed on the disk or volume. To stop the process at any time, click the ***Stop*** button for the particular disk or volume. Click the ***Stop All*** button to cancel wiping for all disks.

4. Verify results

   This is an optional step. Select the wiped volume and click the ***Open in File Browser*** toolbar button to inspect the work that has been completed. The **Active@ KillDisk** scans the system records of the partition, and the Browser tab appears. Existing file and folder names appear with multicolor icons, while deleted file and folder names appear with gray-colored icons. If the wiping process completed correctly, the data residue in deleted file clusters and the space these files occupied in the directory and system records have been removed. You should not see any gray-colored file names or folder names within the wiped volume.

A Processing Summary on page 46 dialog appears when the process is complete. Here you can check the Processing Summary, print labels, certificates, and more.

Note:

Depending on settings, this dialog can be skipped.

Note:

If there are any errors, for example due to bad sectors, these errors will be reported and placed to the log file. If such a message appears you may cancel the operation or continue wiping out disks.

All deleted files and system records on wiped volumes became unrecoverable.

Related information

Disk Wipe on page 76
Processing Summary on page 46
Generated Documents on page 52

Resume Stopped or Interrupted Erase

Disk erasure can be a time-consuming task. Erasing larger disks (10TB+) with sanitizing standards that include multiple overwrite passes can take several hours. If an interruption occurs during erasure (such as a user-initiated stop, disk failure, computer reboot, etc.), the user has the following options:

- **Erase the disk completely** from the beginning;
- **Resume erasure** from the point where it stopped (time-saving option);

When the application starts, all detected disks are analyzed for previously interrupted erasures. If such erasures are detected on one or more disks, the ***Resume Erase*** button becomes active for those disks. Disks with stopped or interrupted erasures are marked with a red **Interrupted Erase** label.

Note:

If disks with interrupted erase being detected after program start, pop up dialog appears automatically suggesting you to ***Resume Erase***. You can run ***Resume Erase*** from here, or select the particular disks later on.

Resume Erase complete process is described below:

1. Commence task

   Select a particular disk or group of disks to launch *Resume Erase* for. Use ***Ctrl+Left Mouse Click*** for multiple selection.

   Initiate the *Disk Erase* task using one of the following methods:

   - Click ***Resume Erase*** command on the action toolbar;
   - Click ***Actions*** > ***Resume Erase*** command from main menu;
   - Click ***Resume Erase*** command from disk's context menu.

2. Configure task options

   After Resume Erase Disk dialog appears, all disks where ***Resume Erase*** option is available will be displayed. You can select more disks for resume erase (if available) or deselect some selected disks.

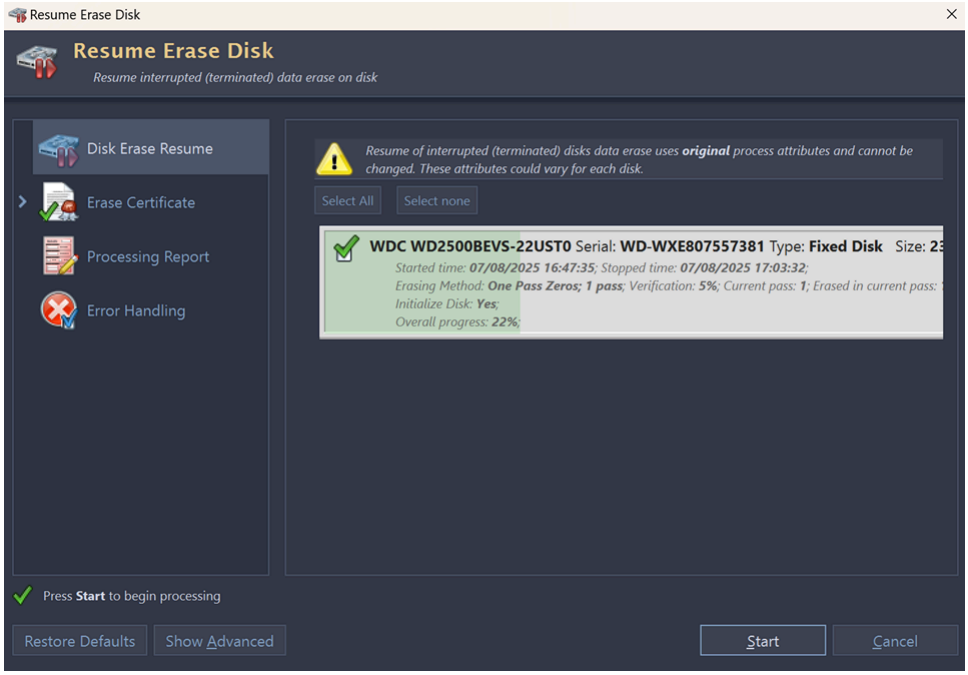


Figure 14: Resume Erase Options

If necessary, configure task options on each page of the Erase Disks dialog, which includes the following:

- Erase Certificate on page 77
- Processing Report on page 80
- Error Handling on page 85

3. Execute task

   Verify selected disks, certificate and report options and click ***Start*** button to resume interrupted erase. Wait until erase is complete.

By default, Processing Summary on page 46 dialog will appear with detailed information of erasure results for each disk, including complete task attributes, log etc.

Related information

Erase Methods on page 186
Processing Summary on page 46
Generated Documents on page 52

Secure Erase

Most of *Solid-state drive (SSD)*) support *Secure Erase (ANSI ATA, SE)* for the low-level purging of all memory blocks on the media. **Active@ KillDisk** is able use SATA *Secure Erase (SSD)* feature and perform fast

unrecoverable erasure. By doing this, you can increase the performance of SSDs for future use. All of the data will be lost without recovery options. Before using this feature make sure user fully understands the Secure Erase concepts on page 185.

Warning:

100% FATAL DAMAGE GUARANTEED TO MEDIA IF THE PROCESS INTERRUPTED (POWER OUTAGE, UNAUTHORIZED SSD EXTRACTION, ETC.)

Make sure your hardware setup is safe from sudden lost of power.

Do not interrupt the process of *Secure Erase* in any manner!

Note:

If there is a need to erase ALL data (existing and deleted) from the hard drive device permanently with sanitation standards (*US DoD 5220.22-M*, *Canadian OPS-II*, *NSA 130-2*, etc.) use Disk Erase on page 35 feature.

Important:

Secure Erase is available for Linux-based packages only (**KillDisk Industrial**, **Active@ KillDisk Linux**, **KillDisk Console** and KillDisk LiveCD in **Active@ KillDisk Ultimate**).

Secure Erase is not available in Windows-based packages, including applications running under Active@ Boot Disk (which is based on WinPE). For security reasons Microsoft intentionally blocked IOCTL_ATA_PASS_THROUGH function in all the latest Windows editions starting from Windows 8.

Secure Erase complete process is described below.

1. Select SSD disks

   Select disks marked as  in ***Local Devices*** view. You may select multiple disks to be erased simultaneously.

2. Start secure erase

Open ***Secure Erase*** dialog using one of the following methods:

- Click ***Actions*** > ***Secure Erase*** command from main menu
- Click ***Secure Erase*** command from disk's context menu

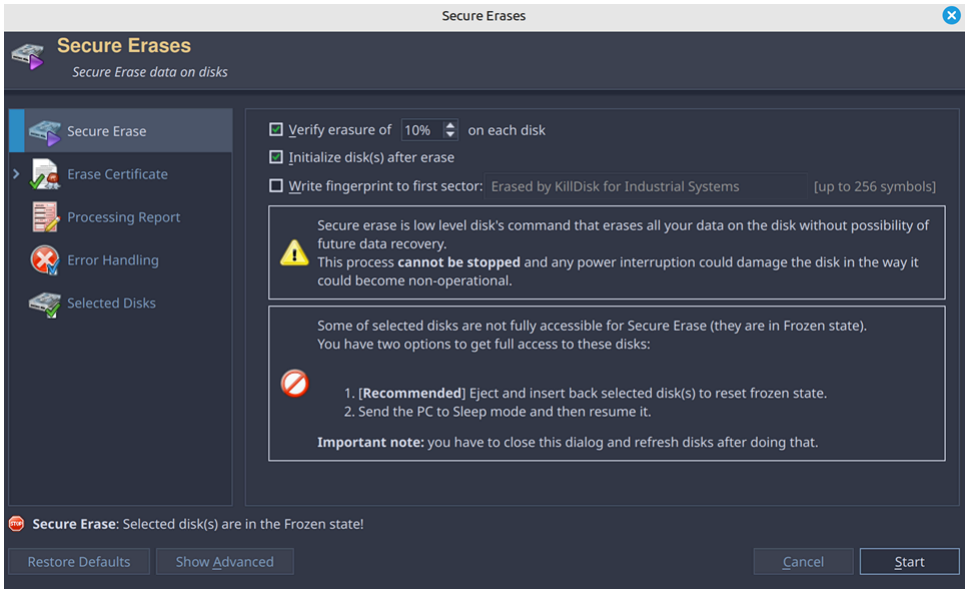


Figure 15: Secure Erase dialog

3. Confirm options

   Use tabbed views to adjust secure erase preferences if necessary.

   Available preferences are:

   - Secure Erase on page 75
   - Erase Certificate on page 77
   - Processing Report on page 80
   - Error Handling on page 85

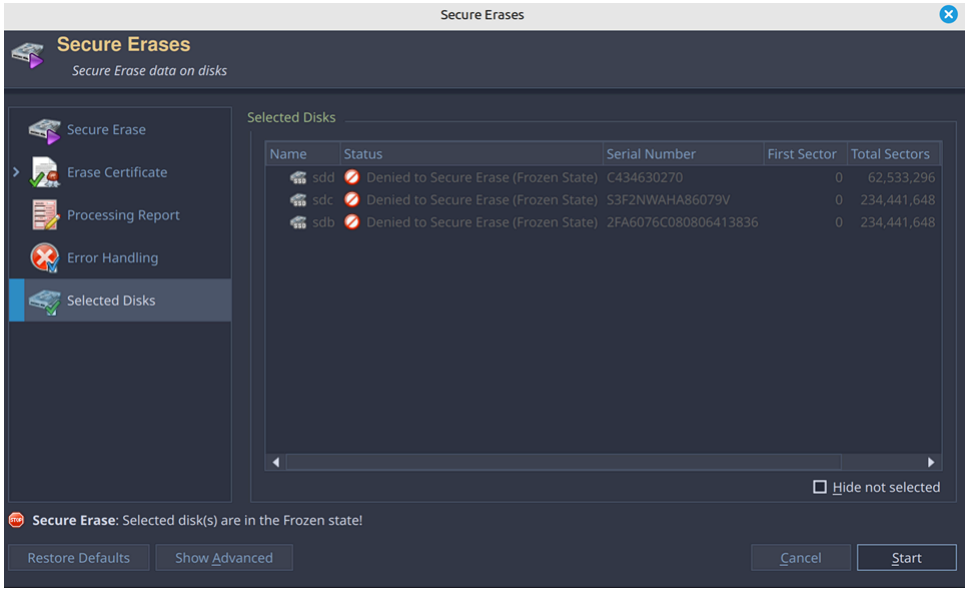
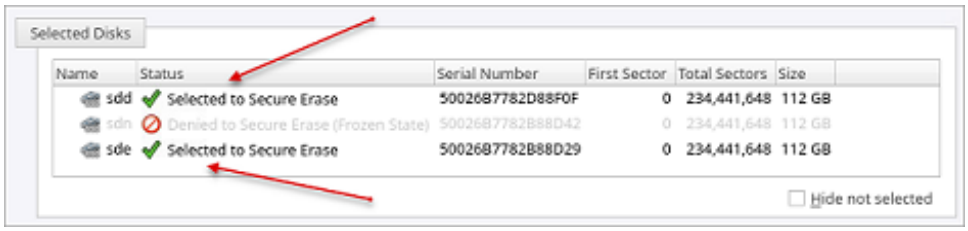


Figure 16: Selected Disks for Secure Erase

Important:

Only disks which state is NOT *frozen SSDs* can be selected for Secure Erase.



Warning:

In case if *SSD which state is Frozen* has been selected for Secure Erase the following message appears:

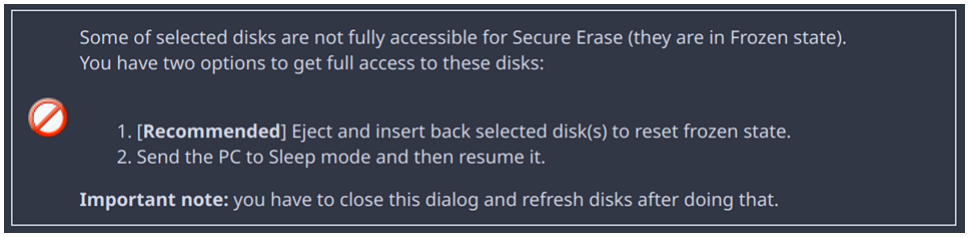


Figure 17: Frozen SSD Warning

You have options either to eject and insert back the SSD, or send PC to Sleep mode and resume it back to get full access to the disk and proceed with a Secure Erase.

4. Start secure erase

   Click ***Start*** button to reach the final step before erasing disk data completely without any possibility to be recovered. Confirm Secure Erase action by typing a predefined keyphrase.

   Click ***OK*** button to confirm erase and start erase process.

5. Observe progress

   There is no progress indicator and Stop action available for the Secure Erase. The feature is implemented inside SSD controller.

   The only time elapsed is available and can be displayed.

After Secure Erase process is completed the Processing Summary dialog appears:

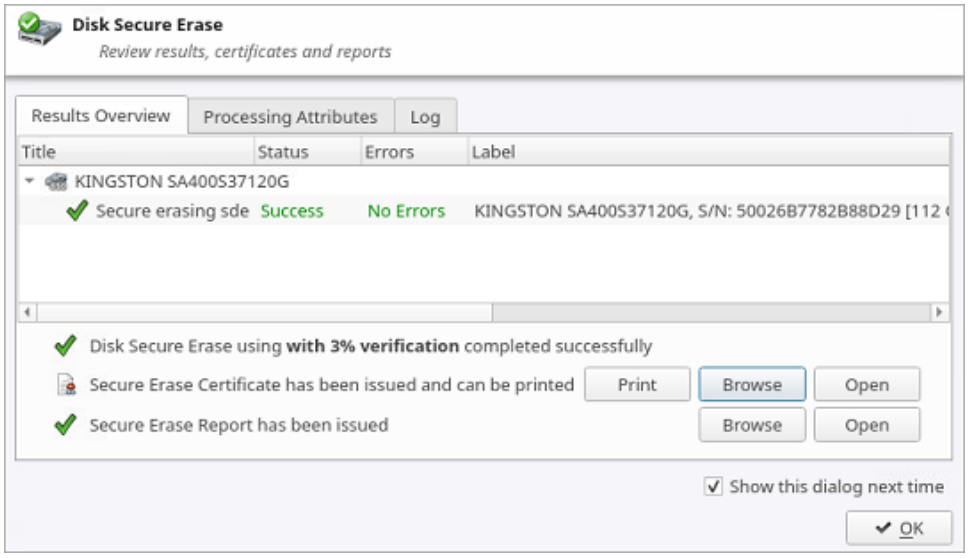


Figure 18: Secure Erase Processing Summary

Now you may ***Print*** and ***Open*** Erase Certificates on page 52 and work with Processing Reports on page 70XML Reports.

If there are any errors they will be reported.

Related concepts

Secure Erase concepts on page 185

Related information

Secure Erase on page 75

Processing Summary on page 46

Generated Documents on page 52

Processing Summary

Once the **Active@ KillDisk** finishes processing tasks, a Processing Summary dialog appears. It contains all information regarding the performed task, including starting attributes, logs, and detailed results. It also provides an opportunity to generate additional outputs (prints) if applicable

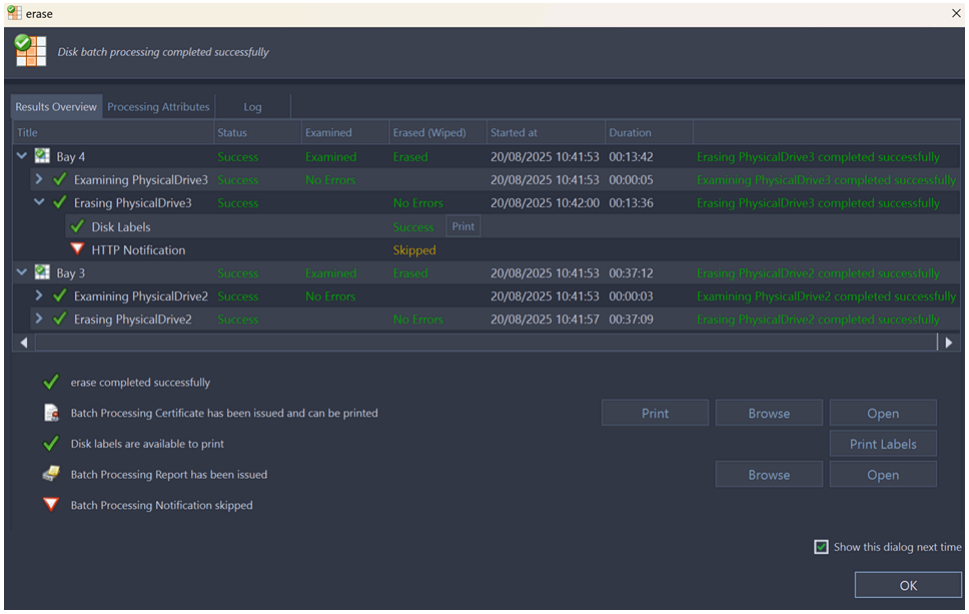


Figure 19: Example of Processing Summary

Title
All the devices processed are displayed with their erase status

Status
An actual erase status (success/fail)

Errors
Displayed number of errors detected (if any)

Label
Volume or partition description

Method
Erase/Wipe sanitizing method being used

Erase Passes
Number of overwriting passes performed

Started at
Time & date of operation's start

Duration
Duration of the operation.

The Processing Attributes tab contains detailed information about operation status and processing attributes.

The Log tab displays only the log records that relate to the completed process, providing detailed information about all aspects of the processing sequence.

Note:

The Wipe operation will produce a similar processing summary for the Disk Wipe.

Disk Certificate
Status of the saved PDF certificate. Allows user to print certificate (***Print*** button), browse certificate directory with a file browser (***Browse*** button) or examine certificate (***Open*** button).

Print Labels
Examine, customize, change options and print Labels by clicking the ***Print Labels*** button.

Disk Processing Report
Status of the saved Processing Reports on page 70. Examine the disk processing report *.xml* file (click ***Browse*** button to navigate to the containing folder) or preview the report (***Open*** button).

Related information

Generated Documents on page 52

Command line interface

Active@ KillDisk can be executed with some predefined settings when started from a command prompt with specific command line parameters.

Active@ KillDisk can be also launched in fully automated mode (Batch Mode) which requires no user interaction.

Active@ KillDisk execution behavior depends on either command line parameters (highest priority), settings configured in interactive mode and stored in settings file (lower priority) or default values (lowest priority).

Command Line Mode

To run **Active@ KillDisk** in command line mode just open a command prompt and go to installation directory.

At the command prompt start **Active@ KillDisk** (Windows version) by typing:

```
KILLDISK.EXE -?
```

In Linux environment, type:

```
./KillDisk -?
```

A list of parameters appears. Their description is in the table below:

Table 1: Command Line Parameters

Parameter	Short	Default	Options
no parameter			With no parameter an interactive screen will appear
-erasemethod=[0-23]	-em=	2	0 - One pass zeroes (1 pass, quick, low security)
			1 - One pass random (1 pass, quick, low security)
			2 - US DoD 5220.22-M (3 passes, verify, slow, high security)
			3 - US DoD 5220.22-M (ECE) (7 passes, verify, slow, high security)
			4 - Canadian OPS-II (7 passes, verify, slow, high security)
			5 - British HMG IS5 Baseline (1 pass, verify, quick, low security)
			6 - British HMG IS5 Enhanced (3 pass, verify, slow, high security)
			7 - Russian GOST p50739-95(2 passes, verify, slow, high security)
			8 - US Army AR380-19 (3 passes, verify, slow, high security)
			9 - US Air Force 5020 (3 passes, verify, slow, high security)
			10 - NAVSO P-5329-26 RL (3 passes, verify, slow, high security)
			11 - NAVSO P-5329-26 MFM (3 passes, verify, slow, high security)
			12 - NCSC-TG-025 (3 passes, verify, slow, high security)
			13 - NSA 130-2 (2 passes, verify, slow, high security)
			14 - German VSITR (7 passes, verify, slow, high security)
			15 - Bruce Schneier (73 passes, verify, slow, high security)
			16 - Peter Gutmann (35 passes, verify, very slow, highest security)
			17 - User Defined Method. Number of passes and overwrite pattern supplied separately
			18 - NIST 800-88 (1 pass zeroes, quick, low security)
			19 - NIST 800-88 (1 pass random, quick, low security)
			20 - NIST 800-88 (3 pass zeroes, slow, high security)
			21 - Canadian CSEC ITSG-06 (3 passes, verify, slow, high security)
			22 - US DoE M205.1-2 (3 passes, verify, slow, high security)
			23 - Australian ISM-6.2.93 (1 pass random, verify, quick, low security)
			24 - NIST 800-88 rev.1 (1 Pass Zeros, quick, low security)
			25 - NIST 800-88 rev.1 (1 Pass Random, quick, low security)

Parameter	Short	Default	Options
			26 - NIST 800-88 rev.1 (3 Passes, slow, high security)
			27 - IEEE Std 2883-2022 Clear (2 passes, verify, slow, high security)
-passes=[1-99]	-p=	3	Number of times the write heads will pass over a disk area to overwrite data with User Defined Pattern. **Important:** Valid for User Defined Method only, otherwise - will be ignored
-verification=[0-100]	-v=	10	Set the amount of area the utility reads to verify that the actions performed by the write head comply with the chosen erase method (reading 10% of the area by default). Verification is a long process. Set the verification to the level that works best for you
-retryattempts=[1-99]	-ra=	2	Set the number of times that the utility will try to rewrite in the sector when the drive write head encounters an error
-erasehdd=[0,1..63]	-eh=		Number in BIOS of the disk to be erased. First physical disk has a zero number. In Linux first disk usually named /dev/sda. In Windows Disk Manager first disk is usually named Disk 0. On older systems (DOS, Windows 9x) first disk is usually named 80h (obsolete syntax is still supported in the parameter)
-eraseallhdds	-ea		Erase all detected disks
-excluderemovable	-xr		Exclude all removable disks from erasing when erase all disks (-ea) option is set
-excludefixed	-xf		Exclude all fixed disks from erasing when erase all disks (-ea) option selected
-excludedisk=[0,1..63]	-xd=		Exclude disk from erasing when erase all disks (-ea) option selected
-ignoreerrors	-ie		Do not stop erasing each time a disk error is encountered. When you use this parameter, all errors are ignored and just placed to the application log
-stopaftererrors=[1,2...]	-ir=		Stop erasing process after specific number of writing errors encountered
-initdisk	-id		Initialize disk(s) after erase
-fingerprint	-fp		Initialize disk(s) and write Fingerprint to first sector after Erase
-computerid=[1,2]	-ci=		1 - Display BIOS ID on the certificate
			2 - Display Motherboard ID on the certificate

Parameter	Short	Default	Options
-clearlog	-cl		Use this parameter to clear the log file before recording new activity. When a drive is erased, a log file is kept. By default, new data is appended to this log for each erasing process. By default the log file is stored in the same folder where the software is located
-logpath=[“fullpath”]	-lp=		Path to save application log file. Can be either directory name or full file name. Use quotes if path contains spaces
-certpath=[“fullpath”]	-cp=		Path to save erase/wipe certificate. Use quotes if path contains spaces
-savetoremovable	-sr		Save application log and certificates to removable storage
-inipath=[“fullpath”]	-ip=		Path to the configuration file (KILLDISK.INI) for loading the advanced settings.
-noconfirmation	-nc		Skip confirmation steps before erasing starts. By default confirmation steps will appear in command line mode for each hard drive
-beep	-bp		Play tune after processing is complete to inform user
-wipeallhdds	-wa		Wipe out unallocated space on all recognized volumes located on all detected disks
-wipehdd=[0,1…63]	-wh=		Wipe out unallocated space on the disk specified by BIOS number
-test=["filename"]			If you are having difficulty with **KillDisk** use this parameter to create a hardware information file to be sent to our technical support specialists. You must specify the name of the file where to store technical information
-batchmode	-bm		Execute in batch mode based on command line parameters and INI file settings (without user interaction, all operations being stored to log file)
-userpattern=[“fullpath”]	-u=		File to get user-defined pattern from. Applied to User Defined erase method. Each line in the file corresponds to the particular pass pattern
-shutdown	-sd		Save log file and shutdown PC after completion
-nostop	-ns		Prevent erase/wipe stop action
-nohidpi			Disable High DPI scaling
-help or -?			Display this list of parameters

Note: Parameters -test and -help must be used alone. They cannot be used with other parameters.

Note: Commands –erasehdd, -eraseallhdds, -wipehdd and -wipeallhdds cannot be combined.

Type the command and parameters into the command prompt console. Here is an example for Windows :

```
killdisk.exe -eh=80h -bm
```

The same in Linux:

```
./KillDisk -eh=0 -bm
```

In this example data on device 80h will be erased using the default method (US DoD 5220.22-M) without user confirmation and application quits (control returns to the command prompt) when complete.

Here is another Windows example:

```
killdisk.exe -eh=80h -nc -em=2
```

The same in Linux:

```
./KillDisk -eh=0 -nc -em=2
```

In this example all data on the first detected disk (which has 'zero' number or 80h) will be erased using US DoD 5220.22-M method without confirmation. After erase completes, processing summary report will be displayed.

Note: In Linux environment to detect and work with physical disks properly **Active@ KillDisk** must be launched under Super User account. So, if you are not a Super User, you should type a prefix ***sudo***, or ***su*** (for different Linux versions) before each command.

After you have typed **Active@ KillDisk** and added command line parameters press ***Enter*** to complete the command and start the process.

Information on how drives have been erased is displayed on the screen when the operation has completed successfully. **Active@ KillDisk** execution behavior depends on either command line parameters (highest priority), settings configured in interactive mode and stored in the settings file (lower priority), or default values (lowest priority).

Related information

Batch Mode on page 51

Batch Mode

Note: This feature is intended for advanced users only

Batch Mode allows **Active@ KillDisk** to be executed in fully automated mode without any user interaction. All events and errors (if any) are placed to the log file. This allows system administrators and technicians to automate erase/wipe tasks by creating scripts (*.CMD, *.BAT files) for different scenarios that can be executed later on in different environments.

To start **Active@ KillDisk** in batch mode just add the `-bm` (or `-batchmode`) command line parameter to the other parameters and execute **Active@ KillDisk** either from the command prompt or from a custom script.

Here is an example of Batch Mode execution with the wipe command:

```
KillDisk -wa -bm -em=16
```

This command will wipe all deleted data and unused clusters on all attached physical disks without any confirmations using most secure Peter Gutmann's method and control returns to the command prompt when erase completes.

If `-ns` (`-nostop`) command line parameter is specified no user interaction is possible after erase/wipe action started. So user cannot cancel the command being executed.

After command execution completed, application returns the following exit codes to the Operating System.

Return codes:

0 (Zero)

KillDisk returns Zero when all disks erased successfully

1 (One)

KillDisk returns One if errors occurred or nothing has been erased/wiped

2 (Two)

KillDisk returns Two if erase/wipe has been completed, but minor warnings occurred

Related information

Command Line Mode on page 47

Generated Documents

Active@ KillDisk maintains the highest standards of disk erasure implementing most modern sanitation methods and provides extensive options for its operations with Erase Certificates on page 52, Processing Reports on page 70 and Disk Labels on page 56 with various Barcodes on page 68.

Erase Certificates

Active@ KillDisk provides printable certificates upon the completion of Disk Erase on page 35, Secure Erase on page 41 or Disk Wipe on page 37. These certificates can be customized to include company-specific information and hardware/procedure description. Configuring custom settings is described in the Erase Certificate on page 77 section of this guide.

Company logo

Company logo can be placed to the certificate instead of the default **Active@ KillDisk**'s logo at the top right corner.

Barcode

A barcode in selected format with encoded tags and attributes for scanning using a barcode scanner.

Company information

Displays all company information provided in the preferences. The user in the sample above only provided a business name. But other company information may also be included in the certificate.

Technician information

Displays the technician information provided in the preferences. This section is for the name of the operator and any notes they may want to include in the certificate report.

Erasure results information

Displays information pertaining to the erasure procedure conducted on the hard drive(s). Type of erasure algorithm, custom settings, date and time started and duration of the erasure are all listed here.

Disk information

Uniquely identifies the disk being erased. Includes information like Name, Serial Number, Size and Partitioning Scheme.

System information
Provides details on the system used to run **Active@ KillDisk** such as Operating System and Architecture type.

Hardware information
Provides details on the hardware used to run **Active@ KillDisk** such as Manufacturer, Number of Processors, etc.

Note:

The system information here only applies to the system running **Active@ KillDisk**, not the system that was erased by the application!

Storing Certificate to PDF

There are options for storing a certificate to file in PDF format as well as encrypting with passwords and digitally signing output PDFs. You can re-print stored to PDF certificates later on, as well as you can validate their integrity and validity.

Certificate location
Save erase certificate as a file in PDF format to the specific location.

File name template
Specify the template for the certificate file name. See the tags available in Appendix tags section.

Encrypt with password
If password field is not empty, output certificate (PDF file) will be encrypted and protected with specified password. This password needs to be typed in any PDF viewer the next time user opens a certificate for reviewing or printing.

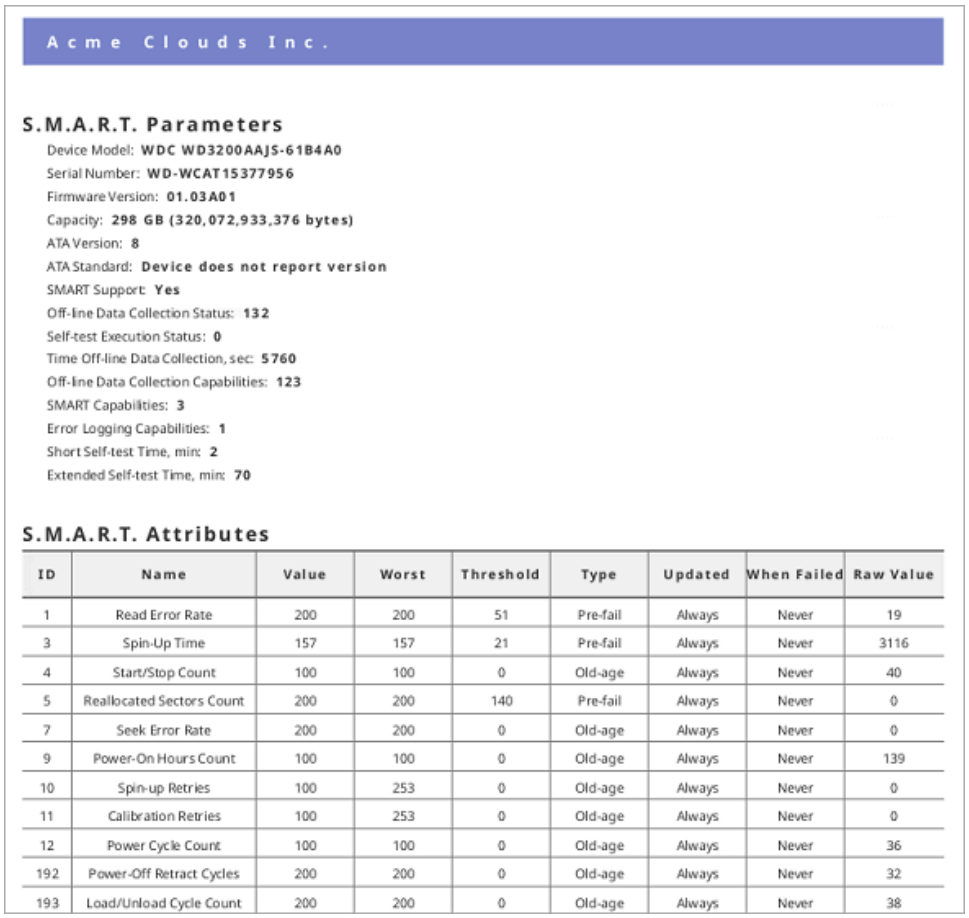
Sign certificate with digital signature
Certificate file (PDF) can be signed with a default digital signature (supplied ***KillDisk.pfx*** certificate) or with your custom digital signature (.PFX file). Digital signature can be verified later on. If Adobe Reader successfully verified PDF document, it is guaranteed that its content hasn't been modified since issue.
If custom digital signature is required, please issue a certificate and specify full path to the custom certificate (.PFX file), as well as .PFX open password (if any) in the related fields.

Display digital signature
Digital signature can be displayed as an overlay text on the first page of certificate. After turning this option on, you can specify overlay text using tags (see tags section), its position on the first page, rectangle dimensions and text size.

Note:

Encrypting certificates with a password and digital signing options are not available when running KillDisk under 32-bit Operating Systems. Only 64-bit platforms supported.

Sample of Erase Certificate



Acme Clouds Inc.

S.M.A.R.T. Parameters

Device Model: **WDC WD3200AAJS-61B4A0**
Serial Number: **WD-WCAT15377956**
Firmware Version: **01.03A01**
Capacity: **298 GB (320,072,933,376 bytes)**
ATA Version: **8**
ATA Standard: **Device does not report version**
SMART Support: **Yes**
Off-line Data Collection Status: **132**
Self-test Execution Status: **0**
Time Off-line Data Collection, sec: **5760**
Off-line Data Collection Capabilities: **123**
SMART Capabilities: **3**
Error Logging Capabilities: **1**
Short Self-test Time, min: **2**
Extended Self-test Time, min: **70**

S.M.A.R.T. Attributes

ID	Name	Value	Worst	Threshold	Type	Updated	When Failed	Raw Value
1	Read Error Rate	200	200	51	Pre-fail	Always	Never	19
3	Spin-Up Time	157	157	21	Pre-fail	Always	Never	3116
4	Start/Stop Count	100	100	0	Old-age	Always	Never	40
5	Reallocated Sectors Count	200	200	140	Pre-fail	Always	Never	0
7	Seek Error Rate	200	200	0	Old-age	Always	Never	0
9	Power-On Hours Count	100	100	0	Old-age	Always	Never	139
10	Spin-up Retries	100	253	0	Old-age	Always	Never	0
11	Calibration Retries	100	253	0	Old-age	Always	Never	0
12	Power Cycle Count	100	100	0	Old-age	Always	Never	36
192	Power-Off Retract Cycles	200	200	0	Old-age	Always	Never	32
193	Load/Unload Cycle Count	200	200	0	Old-age	Always	Never	38

Figure 20: Erase Certificate - 2-nd Page

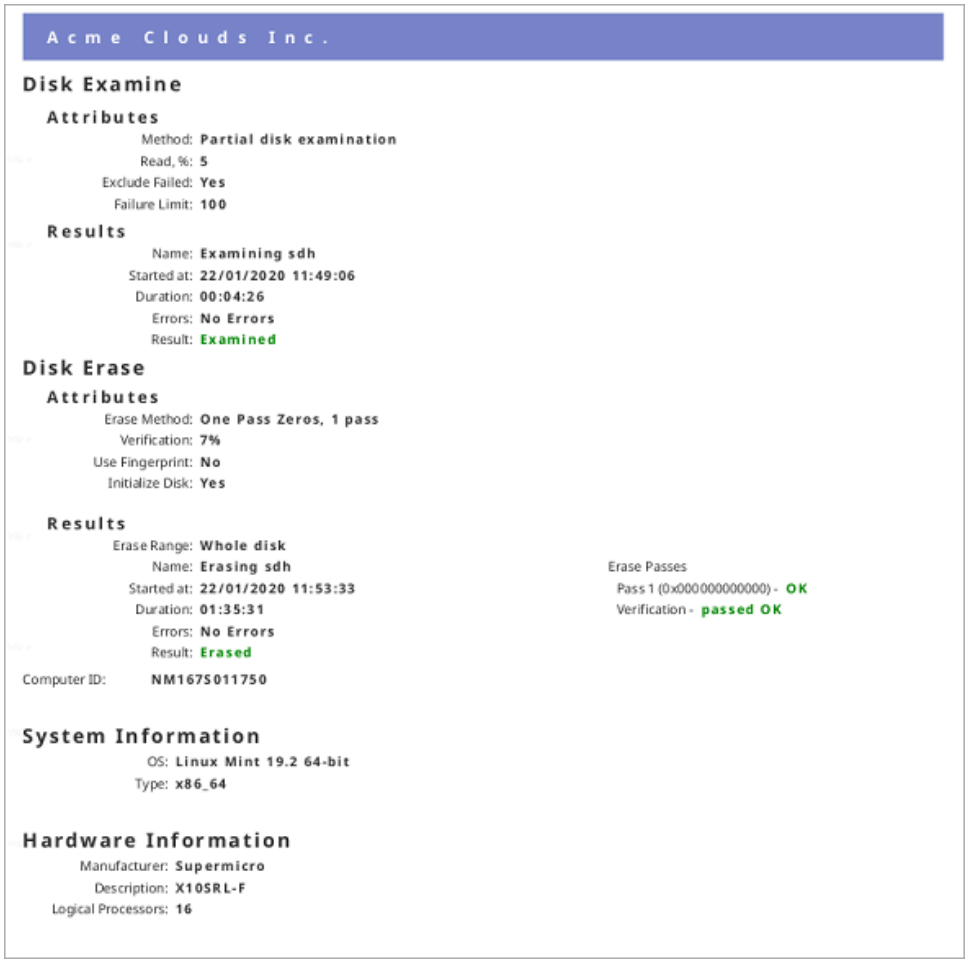


Figure 21: Erase Certificate - 3-rd Page

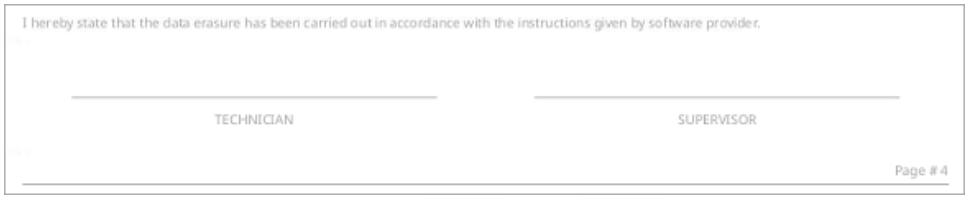


Figure 22: Erase Certificate - Last Page

Sample of Secure Erase Certificate

Figure 23: Secure Erase Certificate - 1-st Page

Related information

Disk Labels on page 56
Processing Reports on page 70
Erase Certificate on page 77

Disk Labels

Along with the Erase Certificates on page 52 certificates **Active@ KillDisk** allows you to print *Disk Labels* issued for each processed disk with final process status and essential disk information . These labels may be completely customizable to print on label tape or on sheet with any dimensions. Simply specify the parameters and **Active@ KillDisk** will prepare the printable labels for you.

Along with the Erase Certificates on page 52, **Active@ KillDisk** allows you to print *Disk Labels* for each processed disk with the final process status and essential disk information. These labels are fully customizable to print on label tape or sheets with any dimensions. Simply specify the parameters, and **Active@ KillDisk** will prepare the printable labels for you.

To organize Disk Label attributes for different use cases, you can use Disk Label Presets on page 57 – a set of label attributes, including the actual Disk Label Templates on page 57 used for different processing scenarios. This preset can be selected on the process configuration page.

At Processing Summary on page 46 dialog Disk Labels can also be printed for each disk in necessary.

Print Labels Dialog

This dialog allows you to configure the labels and prepare them for printing. The top of the dialog shows a list of the drives that will have labels generated for them. At any point in the operation, a sample of the label is shown in the ***Preview*** window on the left side. The right side of the dialog contains the styling and template configuration options.

In case of printing labels on multi-label peel-off sheets, the starting position can be adjusted to select the proper starting row and column.

Once all the settings are configured, you can view the print preview by clicking the **Continue** button. The preview (optional and can be skipped) displays what the print will look like, and from here, the print job can be sent to a printer configured in the system.

Related information

Erase Certificates on page 52
Disk Label Presets on page 81

Disk Label Presets

Disk Label Preset is named set of Disk Label attributes used to for different processing scenarios. These presets can be selected on processing page.

Label Presets can be managed in application preferences on Disk Label Presets on page 81 page.

Disk Label Templates

Disk Label Template defines how content of your label will be placed on paper material: location (offset) and size. One of the predefined Label Templates can be used or custom template can be created. Predefined (standard) supported templates are:

Multi-label peel-off sheets

- Avery 02222;
- Avery 05444;
- Avery 5160;
- Avery 5163;
- Avery 5167;
- Avery 5217;

Tape (label) template

These templates are used for dedicated label printers like Brother QL-700 or Zebra.

- DK-1202;
- DK-1209;
- DK-2205

In addition to predefined templates, a custom template can be created for use with either sheet or tape media: Create a New Label Template on page 84.

Disk Label Templates can be managed on Disk Label Templates on page 83 preference page.

Tape Label Printers

Tape label printers are specialized devices designed to produce adhesive labels on continuous tape rolls, offering a versatile solution for labeling, identification, and organization across various industries. Unlike standard paper label printers, tape label printers use durable, often laminated tape that provides superior

resistance to water, chemicals, abrasion, and environmental factors, making them ideal for long-term labeling applications.

Leading Manufacturers

Several manufacturers dominate the tape label printer market, each offering distinct features and capabilities. Brother is widely recognized for its P-touch series, which ranges from portable handheld models to desktop units suitable for office and light industrial use. These printers are popular for their ease of use, affordability, and wide selection of tape widths and colors. Zebra Technologies specializes in industrial-grade label printers designed for high-volume, demanding environments such as warehouses, manufacturing facilities, and logistics operations. Their printers support various label materials and offer advanced connectivity options for integration with enterprise systems. DYMO, a subsidiary of Newell Brands, produces compact and user-friendly label makers ideal for small businesses, home offices, and retail environments. Epson offers the LabelWorks series, known for reliability and versatility in both professional and consumer applications. Additionally, manufacturers like Brady and Casio provide specialized solutions for safety labeling, facility management, and industrial identification.

Common Use Cases

Tape label printers serve numerous practical applications across different sectors. In IT and data centers, they are essential for labeling network cables, servers, storage devices, and equipment racks, ensuring proper identification and maintenance tracking. Manufacturing and industrial facilities use them for inventory management, asset tracking, safety warnings, and compliance labeling on machinery and products. Healthcare organizations rely on tape labels for specimen identification, medication labeling, patient records, and medical equipment tracking. Retail and warehouse operations utilize these printers for product labeling, shelf organization, shipping identification, and barcode generation. In office environments, tape label printers help with file organization, supply management, and general administrative labeling tasks. The electronics repair and refurbishment industry particularly benefits from tape labels for component identification, warranty information, and quality control documentation on processed devices such as hard drives and computer components.

The durability and professional appearance of tape labels make them an indispensable tool for organizations requiring permanent, weather-resistant, and clearly legible identification solutions. As technology advances, modern tape label printers increasingly offer wireless connectivity, mobile app integration, and compatibility with industry-standard labeling formats, further expanding their utility in today's digitally connected workplaces.

The **Active@ KillDisk** supports major label printers out-of-the-box, including thermal and thermal transfer printers, as well as printers for self-adhesive label sheets.

Add Zebra printer in Linux systems

Add a printer to the Linux system and prepare it for printing labels.

1. Plug printer to computer

   Connect the printer to the computer using the USB interface or to the local network using the network interface.

2. Start printer configuration utility

   Press ***Start button*** > ***Administration*** > ***Printers***.

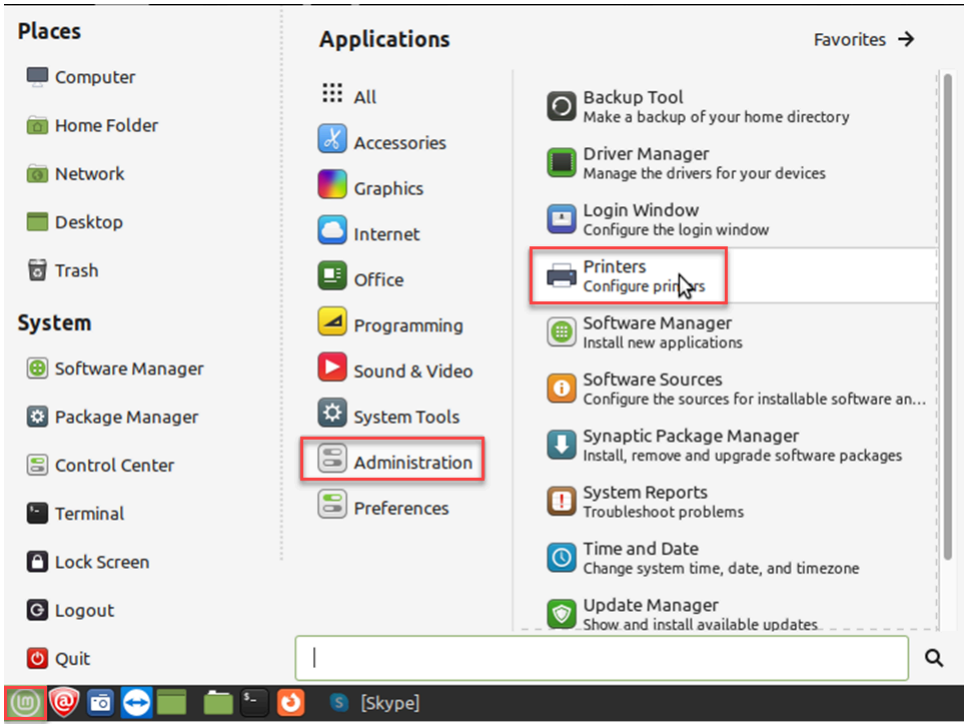


Figure 24: Start Printer application

3. Add the printer

   Press ***Add*** button.

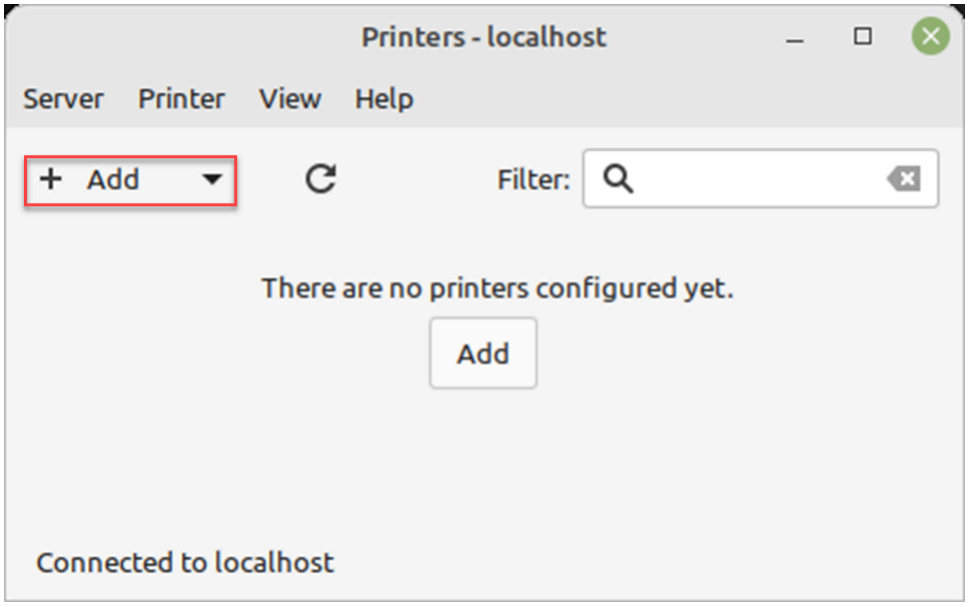


Figure 25: Add new printer

4. Select the device

Choose the Zebra printer in the devices list or enter the location of the network printer

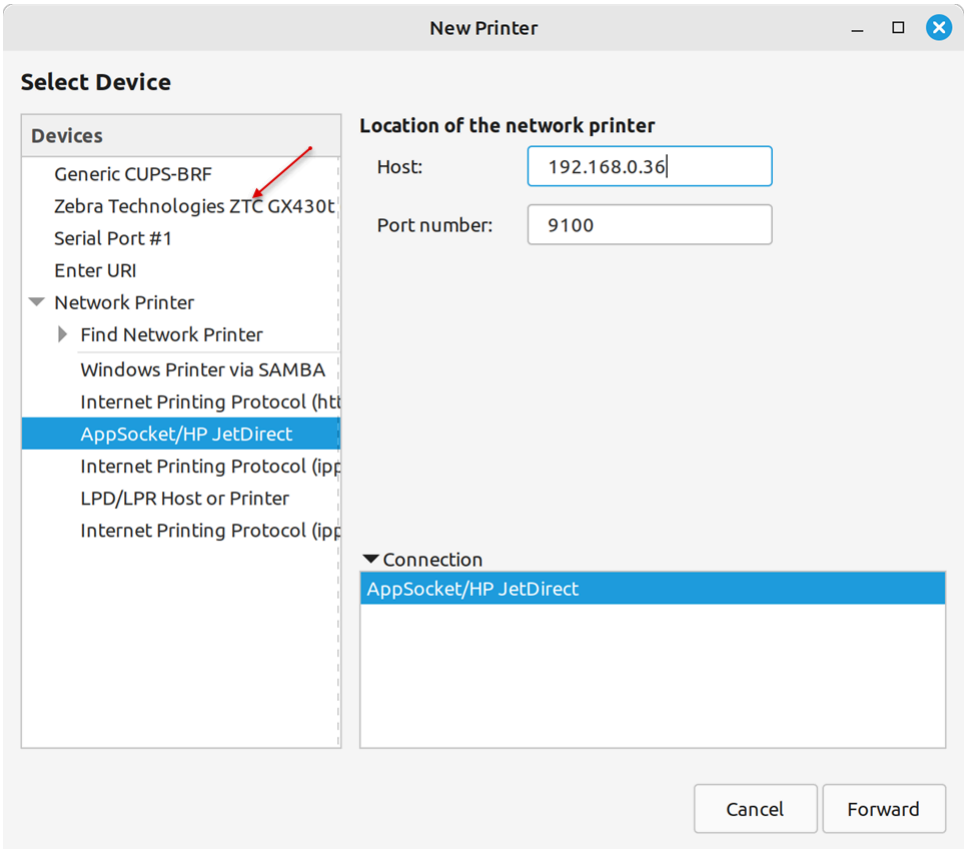


Figure 26: Select Device

- For USB connection, the printer displays in the device list. Select the Zebra printer.
- For network connection, select ***Network Printer*** > ***AppSocket/HP JetDirect*** and enter the network address in the ***Host*** field.

Press the ***Forward*** button for continue.

5. Choose the driver

Select the Zebra printer from the database and press the ***Forward*** button.

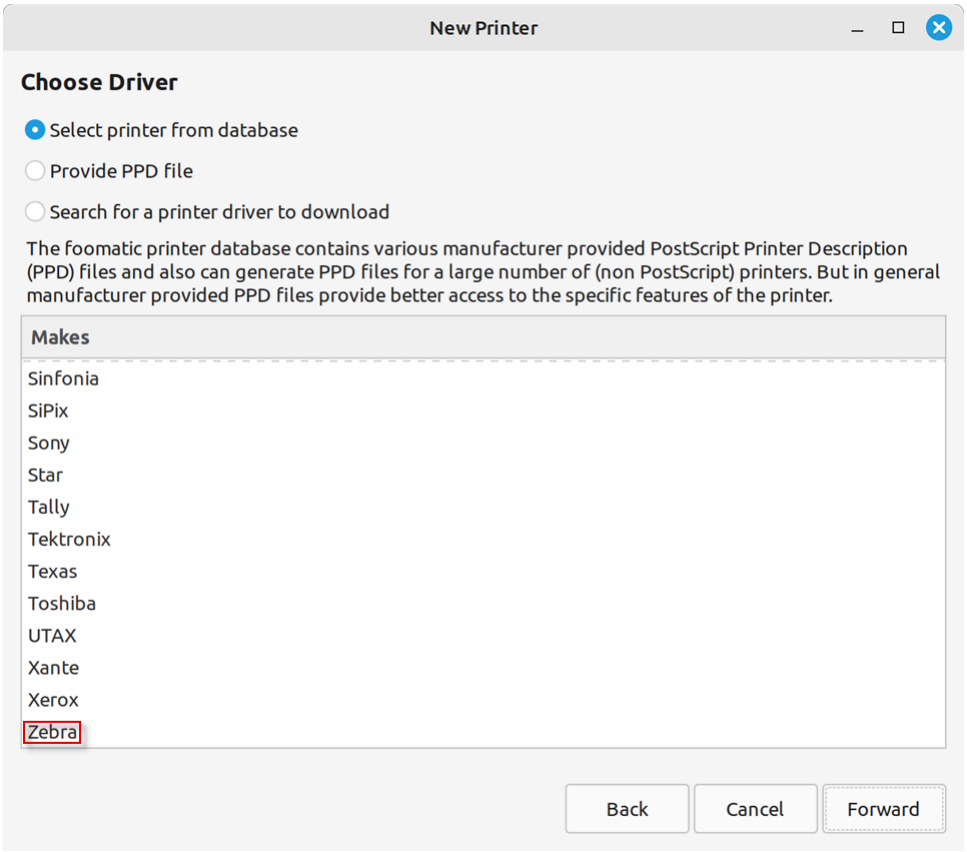


Figure 27: Select printer from database

6. Choose the driver

   Select the model of the printer driver and press the ***Forward*** button.

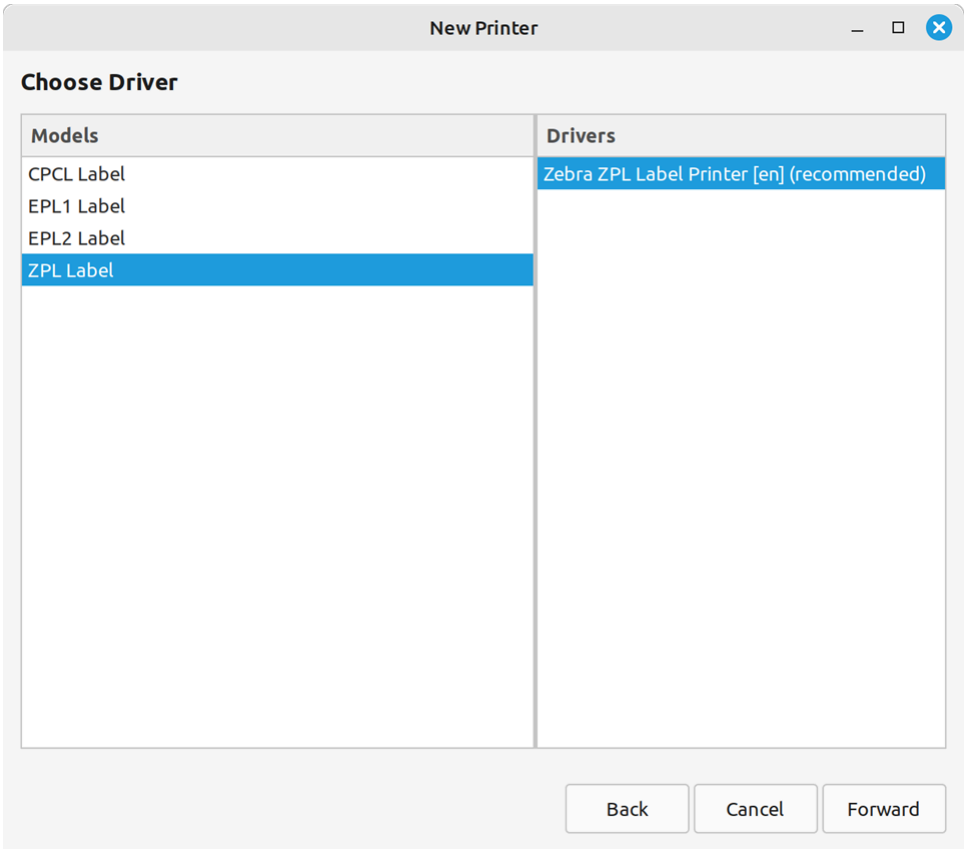


Figure 28: Printer driver model

7. Edit printer description (optional)

   If necessary, edit the printer description and press the ***Apply*** button.

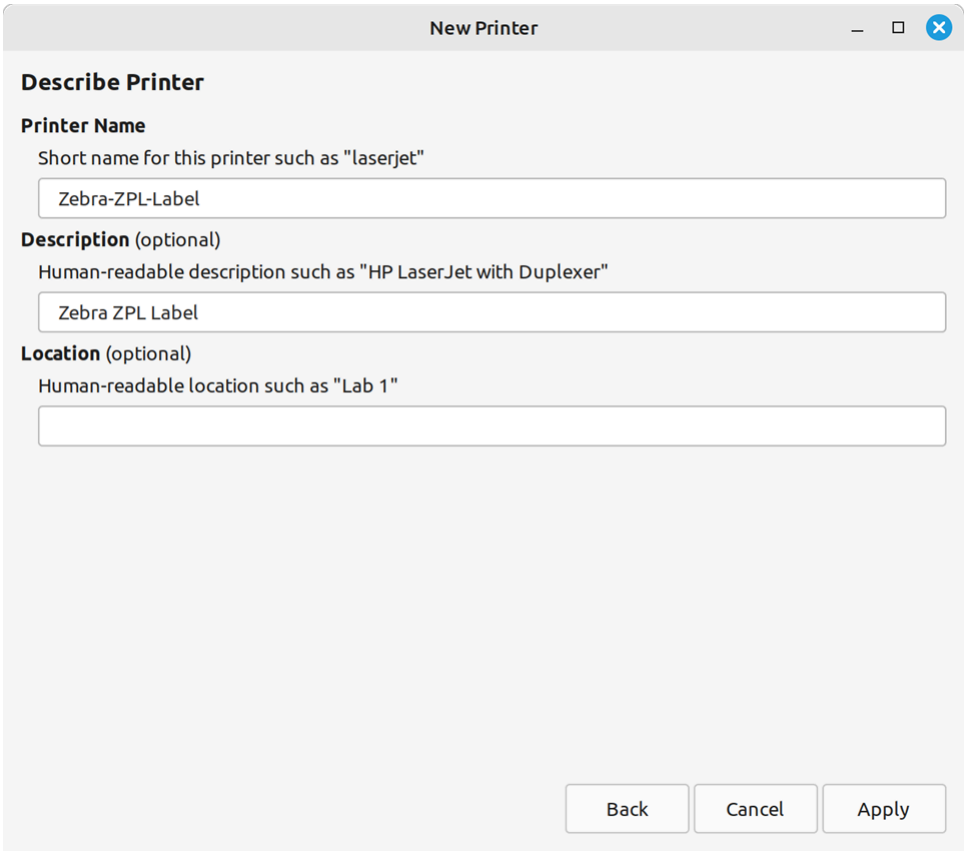


Figure 29: Describe printer

8. Edit printer properties

Double-click on the printer icon to open the printer properties.

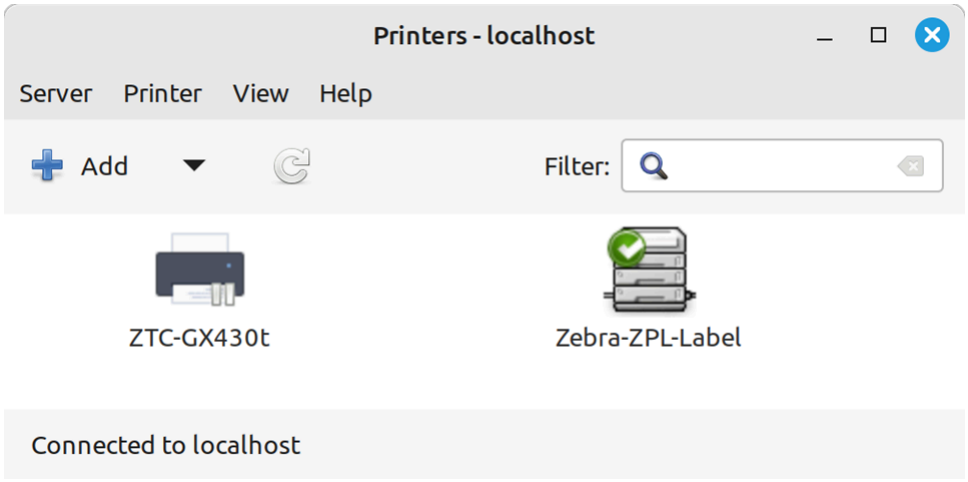


Figure 30: Open the printer properties

Specify the necessary printer attributes for printing as shown in the figure below and press the ***OK*** button to apply the changes.

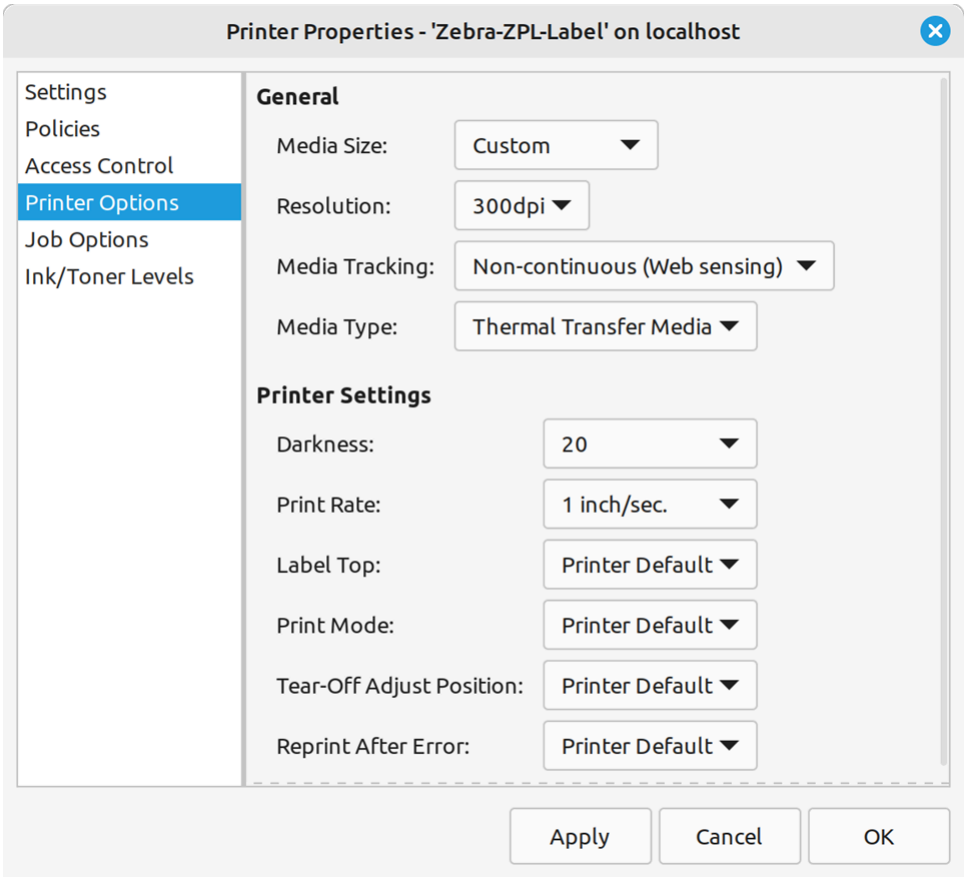


Figure 31: Printer properties

The Zebra printer is now installed and ready to use.

Print test label by Zebra

Set up Active@ KillDisk for printing labels. For correct label printing, need to configure the relevant sections in Preferences.

To do this, proceed as follows:

1. Open Preferences

   Navigate to ***Tools*** > ***Preferences*** or press the ***F10*** shortcut key.

2. Choose label template

   Open the Disk label templates tab and choose the label template which actually corresponds to the label. If the drop-down list doesn't contain the required template, need to create new template.

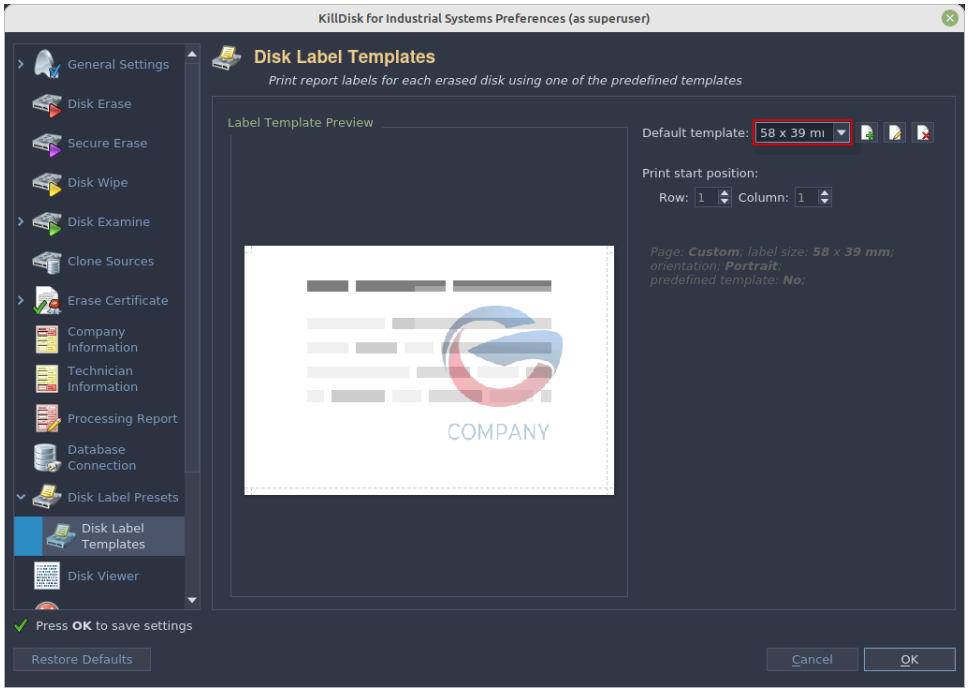


Figure 32: Add new label template

To create a new label preset, proceed as follows:

3. Open Preferences

   Navigate to ***Tools*** > ***Preferences*** or press the ***F10*** shortcut key.

4. Start creating a template

   Select the Disk label templates tab and press the Create new label template icon.

5. Input label parameters

   Enter the label parameters for the new template.

6. Select the template as default

   Open the ***Default template*** drop-down menu and select the new template.

7. Set default printer

   Select the printer for labels from the drop-down list.

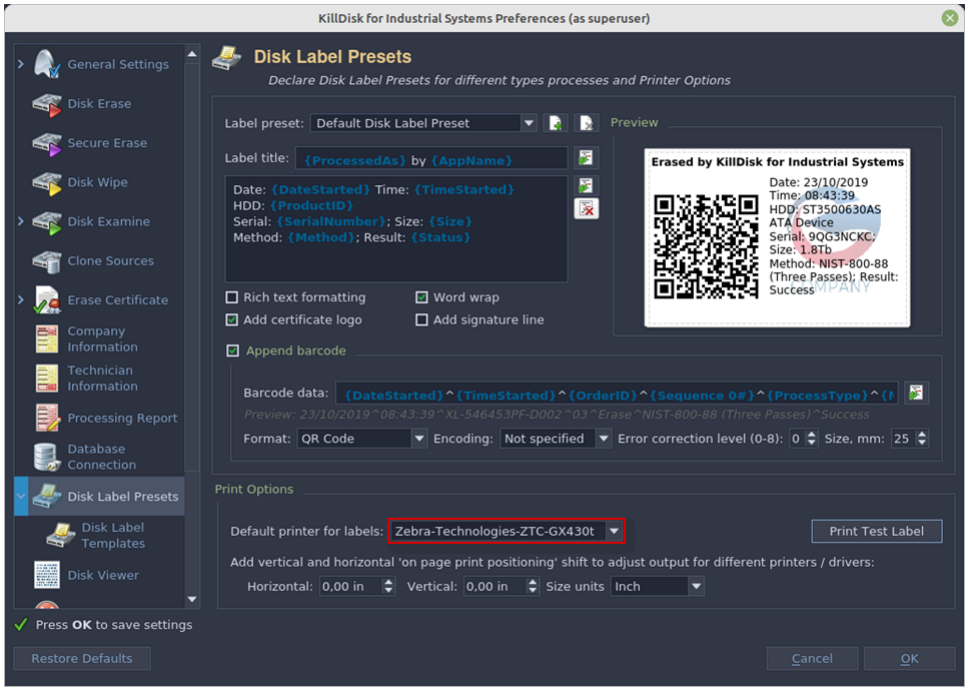


Figure 33: Printer driver model

8. Print test label

   Press the ***Print Test Label*** button and evaluate the result.

The printer and program are ready for label printing.

Related information

Disk Label Presets on page 81
Disk Label Templates on page 83
Create a New Label Template on page 84

Barcodes

Barcodes provide machine-readable data encoding on printed materials, enabling automated identification and tracking throughout disk processing workflows. On certificates and labels, barcodes encode critical information such as drive serial numbers, erasure completion codes, processing dates, and batch identifiers.

The primary advantage of incorporating barcodes into printed outputs is workflow automation. Scanning barcodes eliminates manual data entry, accelerates inventory management, and ensures accurate correlation between physical drives and their digital processing records. For high-volume operations, barcode integration enables rapid verification of erasure status, streamlined chain-of-custody documentation, and efficient integration with enterprise asset management systems. The Active@ KillDisk supports multiple barcode formats to accommodate various scanning infrastructure and compliance requirements.

QR code
Two-dimensional barcode that encodes complex, structured text into high-capacity, machine-readable data, supported by error-correction algorithms that enable reliable scanning even when the code is partially damaged or obscured. Supports **encoding** and **error correction level**.

Aztec 2D barcode
High-density, two-dimensional matrix symbol that stores structured data in a compact, machine-readable format using a central *"bullseye"* locator pattern, enabling fast decoding, high data capacity, and reliable reading even when printed small, displayed at low resolution, or partially degraded. Supports **encoding** and **error correction level**.

Code 39, Code 93, and Code 128 1D barcodes
These are one-dimensional (1D) linear barcode symbology that represents data using variable-width bars and spaces to encode alphanumeric characters, including letters, numbers, and select symbols. It is self-checking, easy to print, and widely used for labeling, inventory, and industrial applications due to its simplicity and broad scanner compatibility.
Code 39 is the simplest and least dense, *Code 93* improves on Code 39 with greater capacity and enhanced error detection, and *Code 128* is the most compact, versatile, and reliable of the three, supporting full ASCII with mandatory checksums.

Feature	Code 39	Code 93	Code 128
Data Density	Low	Medium	High
Character Set	43 chars (A–Z, 0–9, limited symbols)	Full ASCII via shift characters	Full ASCII (native)
Barcode Size	Largest for same data	Smaller than Code 39	Most compact
Checksums	Optional (Mod 43)	Mandatory (C + K)	Mandatory (Mod 103)
Reliability	Moderate	High	Very high
Common Use	Industrial, simple labeling	Improved alternative to Code 39, document tracking	Logistics, shipping, retail, healthcare
Best For	Simple, low-density needs	Higher integrity with moderate density	Maximum efficiency and accuracy

Data Matrix 2D barcode
High-density, two-dimensional matrix symbol that encodes large amounts of alphanumeric or binary data within a compact square or rectangular pattern, using built-in error-correction (ECC 200) to ensure reliable decoding even when the symbol is partially damaged, poorly printed, or displayed at very small sizes.

PDF417 barcode
This barcode is a stacked, linear 2D symbology that encodes large volumes of alphanumeric or binary data across multiple rows using high-density patterns and robust error-correction, enabling reliable decoding for applications requiring compact storage of complex information such as identification documents, transport tickets, and logistics labels. Supports **encoding** and **error correction level**.

Encoding

Each barcode symbology (Code 39, QR Code, Data Matrix, etc.) has its own encoding rules as it determines **how data is translated into the visual structure** of the barcode to ensure that any compliant scanner can read the symbol, regardless of printer, device, or system.

Here the list of supported encoding by Active@ KillDisk:

- ASCII;

- ISO8859-(1 - 16);
- Cp437;
- Cp1250;
- Cp1251;
- Cp1252;
- Cp1256;
- Shift JIS;
- Big 5;
- GB2312;
- GB18030;
- EUC JP;
- EUC KR;
- Unicode Big endian;
- UTF8;

Error Correction Level

Error Correction Level is the amount of redundancy built into a barcode that determines how much damage or distortion the symbol can withstand while still being accurately decoded even if parts of it are damaged, dirty, distorted, or missing. Different barcode types use different error-correction methods (e.g., Reed–Solomon for QR Code, ECC200 for Data Matrix, check digits for 1D codes), but the concept is the same. In Active@ KillDisk error level from 0 (none) to 8 (maximum) can be specified.

Default barcode attributes can be specified in application Preferences on page 72 or defined inline when needed for printed material if applicable.

Related information

Erase Certificates on page 52

Disk Labels on page 56

Processing Reports

Processing Report section allows to configure the XML reports generated by **Active@ KillDisk** after operation is complete.

Active@ KillDisk gives you the option to store XML reports for any major operation it performs (Disk Erase on page 35, Secure Erase on page 41, , and Disk Wipe on page 37) on a disk.

Configure Processing Report on page 80 preferences page in order to get XML reports generated and saved to particular location.

These reports may include detailed information regarding erase processes, such as:

Company Information
- Name
- License
- Location
- Phone
- Disclaimer

Technician Information
- Name
- Comments

System & Hardware Info
- OS version
- Architecture
- Kernel
- Processors
- Manufacturer

Erase Attributes
- Erase Verify
- Passes
- Method
- Verification Passes

Error Handling Attributes
- Errors Terminate
- Skip Interval
- Number of Retries
- Source Lock
- Ignore Write Error
- Ignore Read Error
- Ignore Lock Error

Disks
- Device Size
- Device Type
- Serial Number
- Revision
- Product Number
- Name
- Geometric Information
- Partitioning Scheme

Batches
- Name
- Disks
- Time

Additional Attributes
- Fingerprint Information
- Initialization

Erase Result
- Bay
- Time and Date Started
- Disk Information
- Status
- Result
- Time Elapsed
- Errors
- Name of Operation

This reports, as structured source could be used for import, indexing and overall organize processing history.

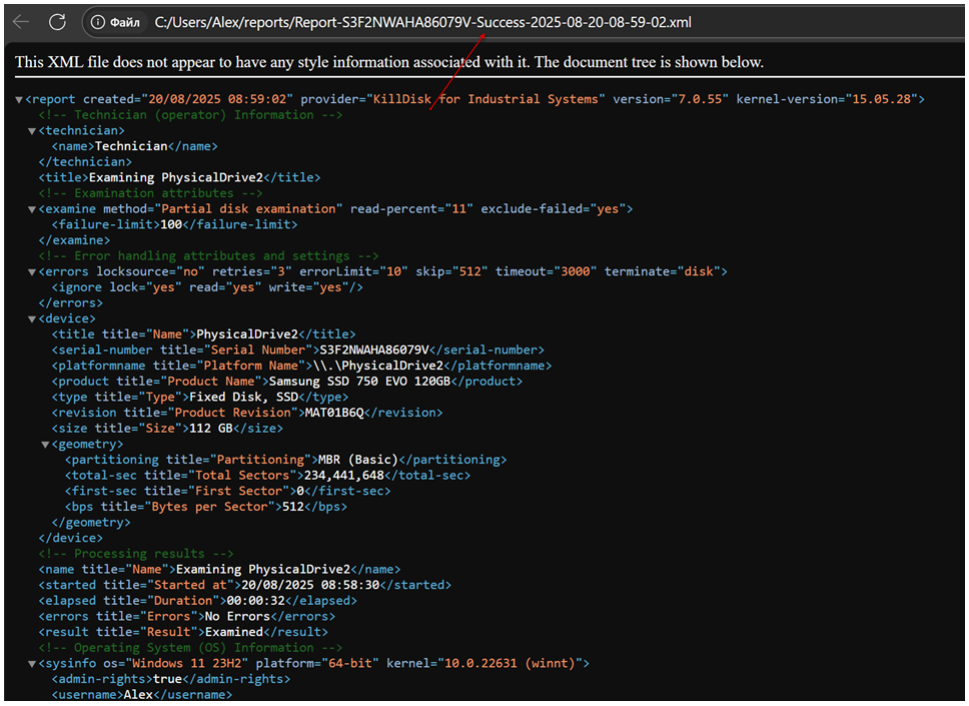


Figure 34: XML Report Sample

Preferences

Active@ KillDisk Preferences dialog is the central location where **Active@ KillDisk** features and settings can be configured.

To open ***Preferences*** dialog:

- From main menu choose ***Tools*** > ***Preferences...***

  or
- Press ***F2*** keyboard shortcut at any time

It's worth mentioning that the Preferences pages reflect the settings from the last executed non-batched process, and any changes to options during task start up are reflected in the Preferences (and vice versa). Read the dedicated topics related to the config pages.

Preferences dialog divided into several tabbed pages:

- General settings on page 73
- Disk Erase on page 74
- Secure Erase on page 75
- Disk Wipe on page 76
- Erase Certificate on page 77
- Company and Technician Information on page 79
- Processing Report on page 80
- Processing CSV Log on page 81
- Disk Label Presets on page 81

- Disk Viewer on page 85
- Error Handling on page 85
- E-Mail Notifications on page 86

Preferences allow to configure all the settings needed for the application proper operation.

General settings

The General page allows to configure general preferences as well as the applications visual and sound representation.

Disk Serial Number

Select default serial number detection method to retrieve the disk serial number by default.

Computer ID Source

Select one (or none) method for how the workstation is identified in logs & reports. Select User-Defined to set a custom workstation name.

Local devices initialization

Specify which devices to initialize at application start up. Disable unnecessary types to improve performance.

Application log

These settings apply to the log file generated by the application. All significant operations performed during the session will be logged.

Log file location

Allows the user to specify where the application log file is saved. By default this is set to a **Active@ KillDisk** installation directory.

File name

This setting configured name for log file. Available save log file with default file name and custom variant of it. Use template for sett custom file name.

Initialize application log when application starts

This setting configures whether the application generates a new log file for each session (erasing the log from the previous session) or appends new sessions to a single log file. Additionally, log files can be named using a specified naming pattern.

Application log detail level

Manipulate the amount of details included in the logs. Options are: ***Minimum*** and ***Maximum***.

Related information

Application Log on page 24

Environment

These are configurable options pertaining to the applications user interface and user experience.

Application style

Configures the color scheme used in the application. Values are: ***Blue***, ***Olive***, ***None (Use OS default)***, ***Silver*** and ***Dark***.

Default toolbars style

Configures how icons are shown in the toolbar.

Default help source

If available, user can select help documentation source to be addressed when requested. Values are: ***PDF***, ***Context Help*** and ***On-line web help.***

Show notification dialog after process complete

Show or hides final process confirmation dialog.

Reset all dialogs

Resets all the settings to the default state.

Sound notifications

These are configurable options related to application sounds: you can use either predefined values or assign your own sounds (User defined sound file).

Use Sound Notifications

Toggles sound tones being used for notifying the user of the completion of a task, errors and notification during an operation: ***Success***, ***With Warnings***, ***With Errors***, ***Failure***.

Action Triggers

Configure actions performed while application is running.

Automatically check for software updates

If this option set, application will check for a new update after every start up.

Action after all processes complete

Select either ***None***, ***Hibernate***, ***Shutdown*** or ***Restart*** system after all running processes completed.

CAUTION:

You will have 30 seconds to abort system hibernation, restart or shutdown.

Export erase certificates and application log to all detected removable media

Upon erase completion all certificates and logs will be automatically exported to attached USB disks (all detected media of removable type).

Disk Erase

The Disk Erase section provides settings configuration for the **Active@ KillDisk** erase procedures.

Erase method

Select the Erase Method to be used for the disk erase procedure. Some of the options below may apply to a specific type of Erase Method.

Read more about Erase Methods on page 186 to determine best applicable method.

Erase verification

Percentage of disk to be verified after disk erasure. The large percentage, the more time it takes to verify written data.

Note:

In some erase methods such as the US DoD 5220.22-M this option is mandatory. After the erase operation has completed this feature will scan the entire drive evenly and verify the integrity of the erase operation. This option

is the percent of the sectors to check across the disk. Most standards specify 10% as an accurate sample size for the verification.

Initialize disk(s) after erase

Writes proper MBR to disk's first sector after erasure complete. This is needed for disk to be visible and accessible by most Operating Systems.

Write fingerprint to first sector

This feature writes the specified fingerprint to the first sector of the erased drive. If erased disk is plugged into the system and system boots from this disk the user will see a message on the screen about the disk being erased by KillDisk.

Print erase labels

This feature prints erase label automatically after erase completion using specific Disk Label configuration.

Erase Confirmation

As an additional precaution, a Confirm Critical Actions on page 75 mechanism is used to prevent accidental data destruction: before each disk erasure, a confirmation dialog appears to provide user-aware protection.

Related concepts

Erase disk concept on page 184
Understanding of underlying mechanisms of data storage organization and data erasure.

Related information

Erase Methods on page 186
Disk Label Presets on page 81

Confirm Critical Actions

As an addition safety measure for critical actions, a key phrase must be entered to proceed in this dialog each time. Three erase confirmation options are available:

- Use keyphrase to confirm erase;
- Use randomly generated keyphrases to confirm erase;
- No keyphrase confirmation.

As a safety precaution to prevent accidental removal of disks' data **Active@ KillDisk** uses the *user-typed keyphrase* mechanism just before the erase procedure is initiated (see below). By default, this precaution mechanism is initialized with the key phrase ***ERASE-ALL-DATA*** . The key phrase can be modified, configured as a randomly generated set of characters or disabled. The keyphrase must be typed correctly in order to start the erase procedure.

See Disk Erase on page 74 page or Secure Erase on page 75 page in application Preferences on page 72 to configure these options.

Secure Erase

The *Secure Erase (SSD)* section provides settings' configuration for the *Solid-state drive (SSD)* specific erase procedures.

Verify erasure

Percentage of disk to be verified after Secure Erase completes.

Initialize disk(s) after erase

Writes proper MBR to disk's first sector after erasure complete. This is needed for disk to be visible and properly accessible by most Operating Systems.

Write fingerprint to first sector

This feature writes the specified fingerprint to the first sector of the erased drive. If erased disk is plugged into the system and system boots from this disk the user will see a message on the screen about the disk being erased by KillDisk.

Erase confirmation

As an additional precaution, a Confirm Critical Actions on page 75 mechanism is used to prevent accidental data destruction: before each disk erasure, a confirmation dialog appears to provide user-aware protection.

Note:

The Secure Erase feature is available in Linux only, not in Windows.

Related concepts

Secure Erase concepts on page 185

Related tasks

Secure Erase on page 41

Disk Wipe

The Disk Wipe section provides settings configuration for Wipe procedure and allows you to specify the erase method to use, verification and a few additional wipe-specific options.

Erase method

Choose one of more than 20 sanitizing methods including many international standards and custom patterns.

Verify erasure

Percentage of disk to be verified after wiping out unused disks' clusters.

Wipe unused clusters

Erase areas of the hard drive that are not formatted and not currently used by the Operating System (data has not been recently written there unless this is a recently deleted partition).

Wipe metadata and system files area

Erase areas on the disk containing information about previous files on the volume. Wiping prevents recovery of files using their remained directory records.

Wipe slack space in file clusters

Erase *File Slack Space* within files. Because files are usually never exactly the size of the space allocated to them there may be unused space within a file that may contain traces of data stored there previously. This algorithm wipes that space to remove these data traces.

Print wipe labels

This feature prints wipe labels automatically after wipe is completed using a specific Disk Label configuration.

Related concepts

Wipe Disk concept on page 190

Related information

Erase Methods on page 186
Disk Label Presets on page 81

Erase Certificate

Erase Certificates section configures options for appearance and storage of certificates in PDF format. If ***Use Erase Certificate*** check box is selected, PDF certificates will be created and available for the immediate printing and storage for future use. Certificates can be customized with Company Information, Technician Information and other information.

Include company information

Use this option to include company's information section to the certificate.

Include technician information

Use this option to include technician's information section to the certificate.

Show KillDisk logo on certificate

Use this option to display a default KillDisk logo at the top right corner of the first page.

Include system info

Ensures that the Operating System specific information for the workstation used for erasure is saved to the certificate, such as:

- Operating system
- Kernel version
- Architecture

Include hardware info

Ensures that the Chassis-specific information for the workstation used for erasure is saved to the certificate, such as:

- Motherboard manufacturer
- Motherboard description
- Number of processors

Include disk SMART information

Use this option to include *S.M.A.R.T.* information section for the disk being erased.

Print options

Always print certificate after disk erase

Prints erase certificate after erase completion automatically.

Skip print preview

Prints erase certificate skipping certificate preview step.

Default printer

Select a default printer for printing erase certificates.

Barcode

If ***Include Barcode*** check box is selected, a barcode section has been added to the certificate in desired format and considering the following options:

Barcode data

Is a string of available tags and attributes concatenated by **^** (*CARET*) delimiter. User is able to compose a custom string with selected values from drop-down list or by simple typing.

Preview

Shows the composed data representation. Barcode data encoded to the actual barcode.

Barcode format

There is a drop-down list of available barcode formats.

Encoding

There is a drop-down list of available encoding schemes for the particular barcode format. The selected encoding is used to encode the barcode data.

Error correction level (0-8)

Affects a size of the barcode. Increasing the level value provides a better scanner readability. Values depend on the barcode format selected.

Related concepts

Name Tags on page 122

Related information

Save Certificate to PDF on page 78
Barcodes on page 68
Processing Reports on page 70
Sign Certificate on page 78

Save Certificate to PDF

Section ***Save to PDF*** offers options for storing a certificate to file in PDF format as well as encrypting it with a password.

Certificate location

Use this option to save erase certificate as a file in PDF format to the selected location.

File name template

Specify the template composed of different tags for the Erase Certificate. See the tags available in Appendix tags section.

Encrypt with password

If password field is not empty, output certificate (PDF file) will be encrypted and protected with specified password. This password needs to be typed in any PDF viewer next time user opens a certificate for printing or previewing.

Related concepts

Name Tags on page 122

Related information

Erase Certificate on page 77
Sign Certificate on page 78

Sign Certificate

Section ***Sign Certificate*** offers options for signing an output PDF certificate with digital signature.

Sign certificate with digital signature

Certificate file (PDF) can be signed with a default Digital Signature (supplied ***KillDisk.pfx***) or with your custom Digital Signature (*.PFX) and can be verified later on. If Adobe Reader successfully verified PDF document, it is guaranteed that its content hasn't been modified since issue.

If custom Digital Signature is required, please issue a certificate and specify full path to the custom certificate (*.PFX file) as well as its open password in the related fields below (***Digital Signature*** and ***Use password to open***)

Display digital signature

Digital Signature can be displayed as an overlay text on the first page of the certificate. After you turn this option on, you can specify overlay text using tags (see tags section) and configure signature position on the first page, rectangle dimensions and text size.

Related concepts

Name Tags on page 122

Related information

Erase Certificate on page 77
Save Certificate to PDF on page 78

Company and Technician Information

The Company Information page contains fields for use in generated outputs where applicable.

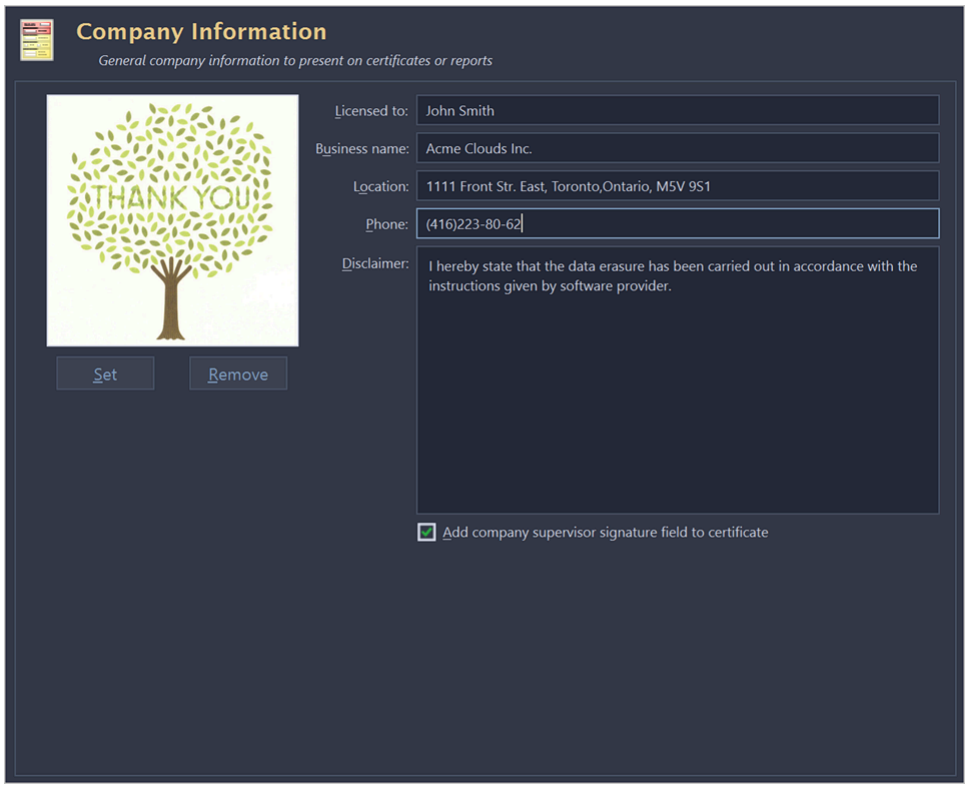


Figure 35: Company Information

To specify a Company Logo image use the ***Set*** button. Select a desired logo image file. Most of the image formats are supported: JPEG, TIFF, BMP and PNG. The logo is previewed in the Company Logo space.

Tip:

It is recommended to use company logo with resolution suitable for printing (300dpi) with a side not exceeding 300px.

Add company's information to the related fields: ***Licensed to***, ***Business name***, ***Location***, ***Phone***, ***Disclaimer***.

When the ***Add company supervisor signature field to certificate*** check box is marked the related field is added to the certificate.

The Technician Information page contains specific technician information for generated outputs where applicable.

Type ***Operator name*** and ***Comments*** to the related fields.

When the ***Add technician (operator) signature field to certificate*** check box is marked the related field is added to the certificate.

Related information

Erase Certificate on page 77
Processing Report on page 80
Disk Label Presets on page 81

Processing Report

This section configures Processing Reports on page 70 attributes:

Report location

Configure where XML erasure reports will be stored. You can use mapped network resource as a storage target.

File name template

Define file name template for the reports, using text and Name Tags on page 122;

Include company information

Adds the company information (defined in Company Information) into the XML erasure report.

Include technician information

Adds the technician information (defined in Technician Information) into the XML erasure report.

Include system info

Ensures that the essential system-specific information is saved in the XML report, such as:

- Operating system
- Kernel version
- Architecture (x86, x64)

Include hardware info

Ensures that the system-specific information is saved in the XML report, such as:

- Motherboard manufacturer
- Motherboard description
- Host (name, domain)
- CPU (logical, physical)
- Memory

Include SMART information for each disk

Adds the information about disk health based on *S.M.A.R.T.* attributes into the XML erasure report.

Note:

If internal tag <task> is present, Results are appeared inside.

Related concepts

Name Tags on page 122

Processing CSV Log

Processing CSV Log section allows to configure CSV (Comma-separated values) log file generated by **Active@ KillDisk** and appending there erase results. First line in CSV log file stores column names: Start Time, Device Name and Serial, Erase Result, etc.

Report log file name

Configure location where CSV log file will be stored. You can use mapped network resource as a storage target.

Save to removable media

Exports CSV log file to all detected removable media (USB Flash Disks).

Record pattern

Define a template for each line in CSV log file. The main tags available are:

Available element:	Tag:
Disk Serial Number	{SerialNumber}
Disk Name	{ProductID}
Erase Method	{Method}
Start Date	{DateStarted}
Start Time	{TimeStarted}
Time Elapsed	{TimeElapsed}
Erase Result (Success/Failure)	{Status}

More tags are available, see the tags section in Appendix.

Related concepts

Name Tags on page 122

Disk Label Presets

Disk Label Presets section allows to adjust label settings for the **Active@ KillDisk**. Labels can be formatted for any printer, page or label type using **Active@ KillDisk** highly customizable labels' formatting features.

General Label Preset attributes

Label Preset

Displays and let you select a default Label Preset or create a new one. Click ***Add New Label Preset*** button  to create a custom label preset with your own specifications. Click ***Delete*** button  to delete the selected label preset .

Label title

Sets a title to be printed (in bold) at the top of the labels. It can be a company name, batch name or any other descriptors you may consider useful to identify the operation. Static text can be typed in or any dynamic attributes (tags) can be inserted at current cursor's position.

Click ***Insert Name Tag*** button  to insert predefined tag from the drop-down list.

Label content

Label's content for the preset. Static text can be typed in or any dynamic attributes (tags) can be inserted at current cursor's position. Click ***Insert Name Tag*** button  to insert predefined tag from the drop-down list. Click ***Clear Pattern*** button to empty all label's area.

Add signature line

Adds a line at the bottom of the label for the technician to sign off on upon completion of the operation.

Add certificate logo

Includes the logo used in the certificate as a label's watermark background.

RTF formatting

use Rich text formatting or plain text.

Word Wrapping

Applies word wrapping to label output.

Barcode options

Selecting ***Append barcode*** check-box will print QR Code or Barcode on the label to be able to be scanned thereafter for third party inventory database.

Barcode data

String including essential erase parameters to be encoded and transformed to QR Code or Barcode. Static text can be typed in or any dynamic attributes (tags) can be inserted at current cursor's position. Click ***Insert Name Tag*** button  to insert predefined tag from the drop-down list.

Preview

Displays a preview of encoded string with the current input settings. Refreshes when any adjustments are made to the settings.

Format

List of supported QR Code and Barcode formats. Currently supported: ***Aztec 2D barcode***, ***Code 39 1D***, ***Code 93 1D***, ***Code 128 1D***, ***QR Code***. Note that different types of Barcodes can accept different size of encoded string.

Encoding

If barcode string contains symbols other than English letters, you can specify encoding (code page) for the particular language.

Error correction level

The lower the error correction level, the less dense the QR code image is, which improves minimum printing size. The higher the error correction level, the more damage it can sustain before it becomes unreadable.

Size, mm

Size in millimeters for the Barcode/QR Code to be printed on the label.

Print options

Define options for label printing including special label printers (Brother QL-700, etc):

Default printer

Define printer to be used exclusively to print labels from the list of installed printers.

Print output adjustments

The print output adjustments section of the dialogue allows you to vertically or horizontally displace the position measured in specific print units to adjust to different printers.

Print test label command let you print Disk Label sample to verify your settings and selected layout attributes.

While changing the options mentioned above, a generic label preview will be rendered in the preview panel, reflecting all settings and refreshing automatically when any adjustments are made.

Related concepts

Name Tags on page 122

Related information

Barcodes on page 68

Disk Label Templates

Disk Label Templates section defines set of **predefined** or **custom** label templates for usage in different scenarios. These templates may be easily selected without opening any additional dialogs. The details of the selected template are displayed below the selection box. If your custom labels differ from any of the templates available, the  button allows you to create a custom template with your own specifications. Additionally, the  button allows you to modify an existing template and the  button deletes the selected template.

The ***Print Start Position*** attributes of the dialogue allows you to select the starting position of the label on the page to print from. The printing won't always start from the 1x1 position, so you can adjust this setting accordingly.

ToCreate a New Label Template on page 84 , click the  button on the Disk Label Templates dialog.

Related concepts

Name Tags on page 122

Related information

Disk Labels on page 56

Create a New Label Template on page 84

Create a New Label Template

In addition to predefined templates, a custom template can be created for use with either sheet or tape media. To create a custom label template, click the ***Create Template*** button on the Disk Label Templates on page 83 page.

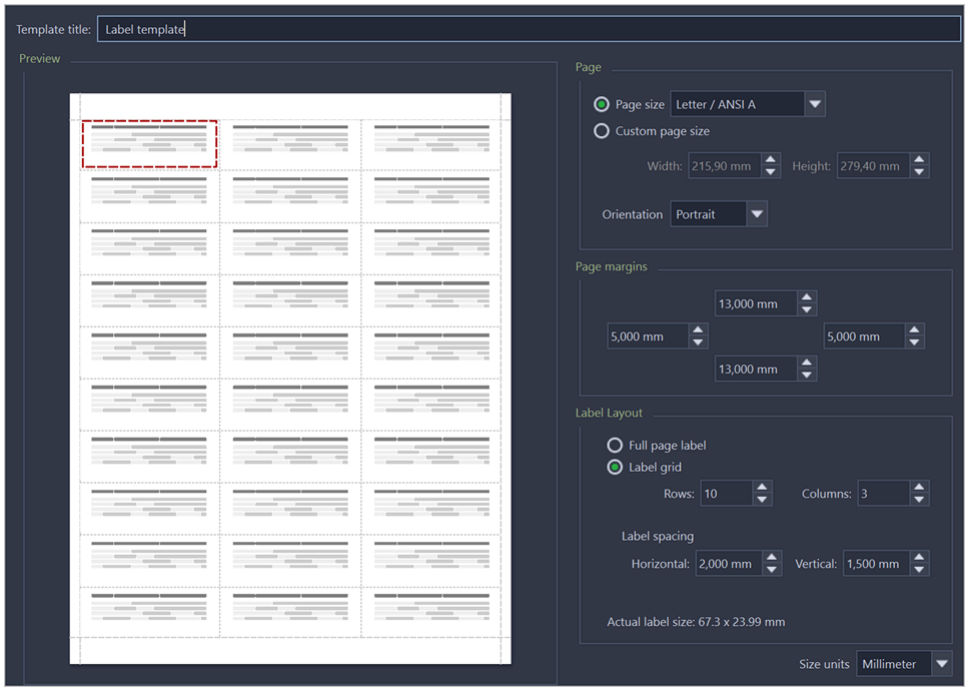


Figure 36: Creating a New Template

Dialog Options

Template title

Sets a custom title for your template. This is the name to refer this template when selecting it in the Print Label dialog.

Page

Specify the dimensions of the page being used to print the labels. Select page size from the list of standard sizes or define custom size using exact measurements. Define page orientation.

Page margins

Page margins are defined for the top, bottom, left and right sides of the page.

Label layout

Define how the labels appear on the page. Define the spacing in between labels on the page and the dimensions of the label grid. Once you entered the proper measurements, **Active@ KillDisk** takes care of all formatting.

Size units

The units of measurement may vary between millimeters, inches, pixels and points. If a value is entered in one measurement and then size unit is changed, the appropriate conversion takes place.

Related concepts

Name Tags on page 122

Related information

Disk Label Presets on page 81
Disk Label Templates on page 83

Disk Viewer

Disk Viewer section allows to set hexadecimal view settings, font and user interaction parameters.

Hexadecimal offsets

Toggles offset format between decimal and hexadecimal.

Lines to scroll

Number of lines to scroll for a single mouse wheel sweep.

Pages to scroll

Number of pages to skip for a single ***Page Up*** or ***Page Down***.

Show ASCII column

Toggles display content in ASCII format.

Show UNICODE column

Toggles display content in UNICODE format.

Bytes per line

Defines amount of bytes per line in hexadecimal display.

Font name

Select any mono-space font from the list of available ones for better view experience.

Font size

Font size to be used in hexadecimal display.

Error Handling

Error Handling section has the advanced settings to configure error handling during long going disk processing or other commands.

Active@ KillDisk allows you to configure error handling during long-running disk processing or other commands, such as:

Abort entire group processing

If erase Batch is in progress and one of the disks has errors, the erase process for ALL the disks in the Batch will be terminated.

Abort only failed disk from group processing

This is the default setting. Failed disks return an error and terminate the erase process. Other disks in the Batch will continue current operation.

Ignore error for group processing

Ignores the read/write error and continues erasing whatever is possible on the disk. None active or forth going operations will be terminated.

Terminate process after number of errors
Sets the error threshold to a certain amount before the disk operation is terminated and deemed unsuccessful.

Number of read/write attempts
Sets the number of attempts **Active@ KillDisk** makes to perform an operation when an error is encountered before it stops execution.

Ignore preceding results
Errors (if any) on previous steps (i.e. Examination) are ignored and following steps (i.e. Erase, Clone) will be executed. If turned off the errors on previous steps will stop all further actions.

Use disk lock
Locks disks from being used by any other applications while operation is in progress.

Ignore disk lock errors
Errors encountered with **Active@ KillDisk** not being able to access locked disks will be ignored.

Ignore read/write errors
Toggle whether read errors or write errors will be just ignored.

Rely upon disk performance
Sets a minimum acceptable read/write speed in megabytes per second for disks to flag under-performing drives.

Related information

S.M.A.R.T Monitor view

E-Mail Notifications

E-mail Notifications sections allows to configure how client can be notified after operation is complete. **Active@ KillDisk** can deliver results of its sanitation process (certificates, reports, logs) by e-mail.

Send to

Type e-mail address where erasing/wiping reports will be sent to.

E-mail attachments

Certificate, XML Report or Log File can be emailed, just mark the related check box.

Note:

E-mail notifications feature is accessible in commercial packages only.

When you mark ***Use E-Mail Notifications*** check box, the ***SMTP Server Settings*** section becomes accessible for the configuration.

SMTP Server Settings

These settings allow to configure mailer settings for delivering erasing/wiping reports to user's mailbox. Simple Mail Transport Protocol (SMTP) is responsible for transmitting e-mail messages and SMTP Server settings need to be configured properly.

Account type

Active@ KillDisk offers you a free SMTP account located on **www.smtp-server.com** that can be used for sending reports out. By default all the required parameters are filled up and configured properly. If your corporate policy does not allow using services other than its

own you need to switch this option to the Custom Account and configure all the settings manually. Ask your system/network administrator to get these parameters.

From

Type e-mail address which you expect these reports to come from.

Connection type

Select encryption type to use: ***No encryption***, ***SSL*** or ***TLS***.

SMTP server

Active@ KillDisk offers you the use of smtp-server.com for a free SMTP account. This account is pre-configured for **Active@ KillDisk** users. Ask your system/network administrator to get the proper SMTP server domain to be used.

SMTP port

For the free SMTP account **Active@ KillDisk** allows you to use smtp-server.com on port 80. This is a standard port being used by all web browsers to access the Internet. This port most likely is open on a corporate and home networks. Other ports can be filtered by and restricted by a network firewall. Ask your system/network administrator to set up a proper SMTP port for the custom SMTP server.

SMTP server authorization

To avoid spam and other security issues some SMTP servers require each user to be authorized before sending e-mails. In this case proper Username and Password required to be typed. Ask your system/network administrator to get proper authorization settings.

Selected Disks

On this page all task subject selection can be reviewed and modified if necessary.

Processing Report section allows to configure the XML reports generated by **Active@ KillDisk** after operation is complete.

Select All

Select all disks' space.

Select Live Volumes

Select only the disk space where live volumes are located.

Select Unallocated space only

Select only the disk unallocated area (the space where no live volumes exist).

Hide not selected

Hide all unselected objects.

Verification and Proof of Disk Erasure Using Hexadecimal Viewer

Scenario: A data center administrator must provide documented proof that decommissioned hard drives containing sensitive financial records have been completely sanitized before disposal or reuse.

Challenge: Regulatory compliance (such as GDPR, HIPAA, or SOX) requires verifiable evidence that data has been irreversibly destroyed. Simply running an erasure process isn't sufficient—auditors need tangible proof that no recoverable data remains on the storage media.

Solution: **Active@ KillDisk** integrated low-level hexadecimal viewer provides direct access to raw disk sectors, enabling administrators to verify erasure at the most fundamental level:

1. **Pre-Wipe Verification:** Before erasure, the administrator uses the hex viewer to examine sectors containing sensitive data, capturing screenshots showing readable file headers, text strings, or identifiable patterns.
2. **Post-Wipe Validation:** After completing the erasure process, the hex viewer displays the same disk sectors, now filled with the specified overwrite pattern (zeros, random data, or DoD 5220.22-M patterns). No remnants of the original data remain visible.
3. **Documentation for Compliance:** The administrator exports hex dump reports showing specific sector addresses and their contents, providing auditors with concrete evidence that data has been completely overwritten at the binary level.
4. **Spot-Checking Multiple Locations:** By examining random sectors across the entire disk – beginning, middle, and end – the administrator confirms uniform erasure throughout the storage device, not just in specific areas.

Result: The hexadecimal viewer transforms **Active@ KillDisk** from a simple erasure tool into a comprehensive verification system, providing the forensic-level proof required for regulatory audits, legal proceedings, and internal security policies. Administrators can confidently certify that sanitized drives contain no recoverable data remnants.

Introduction

The Disk Viewer gives the direct access to raw data stored on physical storage devices at the most fundamental level – the binary or hexadecimal representation of bytes. Unlike standard file operations that work through the operating system's file system layer, low-level editing provides direct access to the underlying data structures, enabling work with individual bytes, sectors, or clusters on a disk, partition, or within files. In Disk Viewer all these information presented in hexadecimal, ASCII or Unicode formats with easy direct and logical navigation allowing user to evaluate the actual content and integrity of:

- Physical Disk
- Logical Drive or Partition
- File

Use Disk Explorer on page 23 to open any of these data storage objects from menu (shortcut is ***Ctrl-H***) or other UI widgets that gives an access to system data structure.

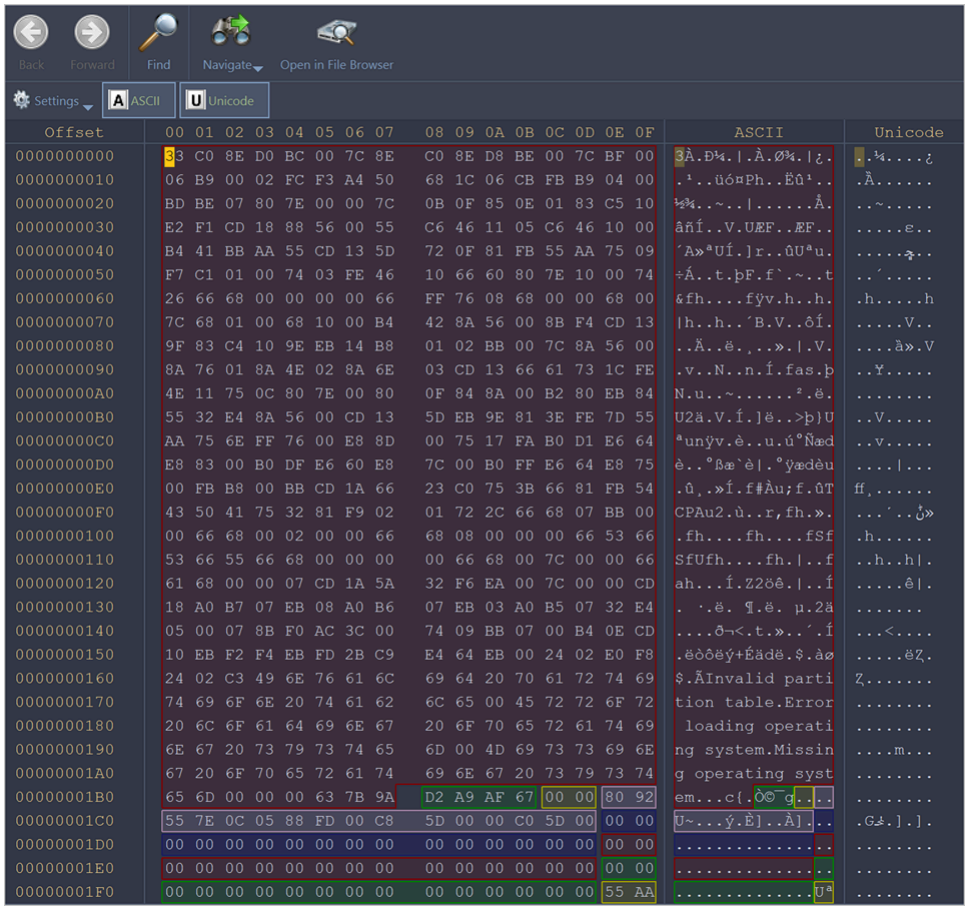


Figure 37: Physical Disk in Disk Viewer

Navigation

After you have opened an object with the Disk Viewer, you may navigate by scrolling block by block, or by jumping directly to specific addresses. You may jump to disk system records such as the boot sector (primary and copy) or a partition table.

Read Navigation and Information on page 90 articles for more.

Data Selection

In order to select data in the ***Hexadecimal, ASCI or Unicode*** area, click and hold down the left mouse button and start dragging to select an area. The selected area background will be highlighted. Release the mouse to finish selecting. You can select an area bigger than will fit into the screen by dragging the mouse beyond the top or bottom edge of the hex editor window.

The alternative way to make a selection is to define a beginning and an end of the block. This method might be more convenient when a large area has to be selected in order to simply select data in a particular range. Move the cursor to the position where you want the selection to start and do one of the following:

- Select the menu command ***Edit*** > ***Beginning of block*** from the ***Edit*** menu in the toolbar;
- Select the menu command ***Advanced*** > ***Beginning of block*** from the ***Advanced*** menu in the edit pane toolbar;
- Right click and select ***Beginning of block*** from a context menu;

- Press ***Ctrl+1*** .

Move the cursor to the end of the desired selection and set the end of a selection in a similar way. If you need to select all the data, you can use the Select All command instead.

Use left mouse button double-click inside the sector for select an entire sector.

Use ***Shift+Beginning of block*** for select from beginning of block to current cursor position.

Use ***Shift+End of block*** for select from current cursor position to end of block.

To select you can also use ***Up, Down, Left, Right, Home, End, Pg Up, Pg Dn*** keyboard buttons in combination with the ***Control*** and ***Shift*** buttons.

Working with the clipboard

Select an area of data as described above and either select:

- the command ***Edit*** > ***Copy*** ;
- the ***Copy*** button from the edit pane toolbar;
- the command ***Copy*** from a context menu;
- press ***Ctrl+C*** .

The selected area will be copied into the clipboard in binary format. If you later want to insert it into a text editor, use the ***Copy Formatted*** command or press ***Ctrl+Shift+C*** instead. It will copy data as a formatted text.

When you copy selected text from the edit pane to the clipboard, you may paste it in text editor in one of three formats:

- Binary – hexadecimal representation of selected data
- Text – text representation of selected data
- Formatted text – formatted hexadecimal and text representation of selected data (as it appears in the editor)

Note:

Please note that you can copy a maximum of 1MB of data into the clipboard.

Related information

Navigation and Information on page 90

Navigation and Information

Basic Navigation

After you have opened an object with the Disk Viewer, you may navigate by scrolling block by block, or by jumping directly to specific addresses. You may jump to disk system records such as the boot sector (primary and copy) or a partition table.

Use the ***Navigate*** button in the toolbar to jump to a specific area in the open object.

When you navigate to an access point through the ***Navigate*** menu or jump to a specific offset or sector, those addresses are stored in a stack. You can move backward and forward to the previous locations by using the ***Back*** and ***Forward*** commands located in the ***Toolbar*** .

The selections that appear depend on the type of object that you are editing.

Direct Navigation

No matter what object is opened for editing, the first two menu items in the ***Navigate*** menu will be ***Go to Offset*** and ***Go to Sector***.

Read Move to Offset on page 94 and Move to Sector (cluster) on page 94 articles for more information.

Logical Navigation

After you have opened an object with Disk Viewer, you may navigate by scrolling block by block, or by "jumping" directly to specific addresses. You may jump to disk system records, such as the boot sector (primary and copy) or partition table. In a file's cluster chain list, you may jump to the first cluster of a continuous cluster chunk when working with a file.

To open the ***Navigate*** menu:

- From the main menu choose ***Actions*** > ***Navigate*** content;
- From the Disk Viewer toolbar, open the ***Navigate*** drop-down menu.

The selections that appear depend on the type of object that you are editing.

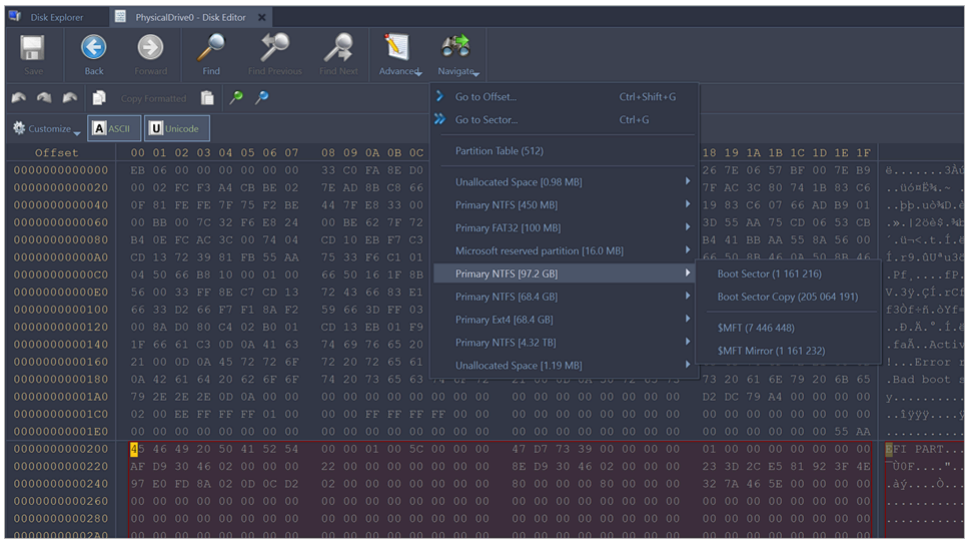


Figure 38: Example. Navigate Menu Selections

Use Property view and SMART Info for detailed information about subject attributes - Property View on page 116.

Related information

Move to Offset on page 94
Move to Sector (cluster) on page 94
Navigate a Physical Disk on page 92
Navigate a Logical Drive on page 93

Navigate a Physical Disk

To navigate to the disk system records of a *Physical device*, click on the ***Navigate*** button in the toolbar. Depending on the partition scheme and contents of the physical disk you are editing, the ***Navigate*** menu will contain different options.

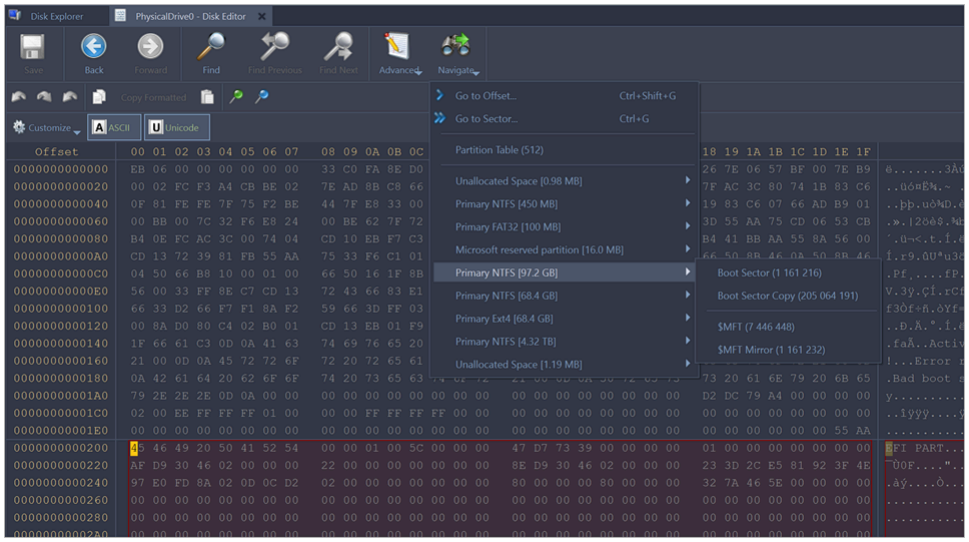


Figure 39: Example. Navigate Menu Selections

Navigating basic disks

After the ***Go to Offset*** and ***Go to Sector*** items there is a ***Partition table*** menu item which allows jumping to *Sector* 0 of a physical disk. As you jump to the partition table, a *Master Boot Record (MBR) Templates (Patterns)* is automatically selected.

If the disk is not empty, the names of the *Partition*s and their system areas will be in sub menus below the ***Partition Table*** menu item.

Navigating dynamic disks

For *Dynamic Disk*s the following system areas are available for direct access:

- *LDM* Private Header
- LDM Primary TOC Block
- LDM Backup TOC Block
- LDM VMDB Block
- LDM KLog
- LDM First VBLK Block

After each access point a sector number is specified in the brackets.

Related information

Navigation and Information on page 90
Move to Offset on page 94
Move to Sector (cluster) on page 94
Navigate a Logical Drive on page 93

Navigate a Logical Drive

To navigate to the disk system records of a logical drive, click on the ***Navigate*** button in the toolbar.

Depending on the *File system* present in a *Logical drive*, the navigation menu will have different access points.

- *Boot Sector*
- Boot Sector Copy
- *MFT or MFT records (Master File Table)*
- $MFT Mirror
- Arbitrary MFT record
- Boot Sector
- Boot Sector Copy (FAT32 only)
- *FAT1*
- FAT2
- *Root directory*
- Boot Sector
- Boot Sector Copy
- FAT
- Root Directory
- *Volume Header*
- Volume Header Copy
- *Superblock*
- Superblock
- Superblock

Some of the access points when used automatically select a corresponding *Templates (Patterns)*. For example, if a boot sector access point is selected, a *Boot Sector* template is applied to the boot sector offset.

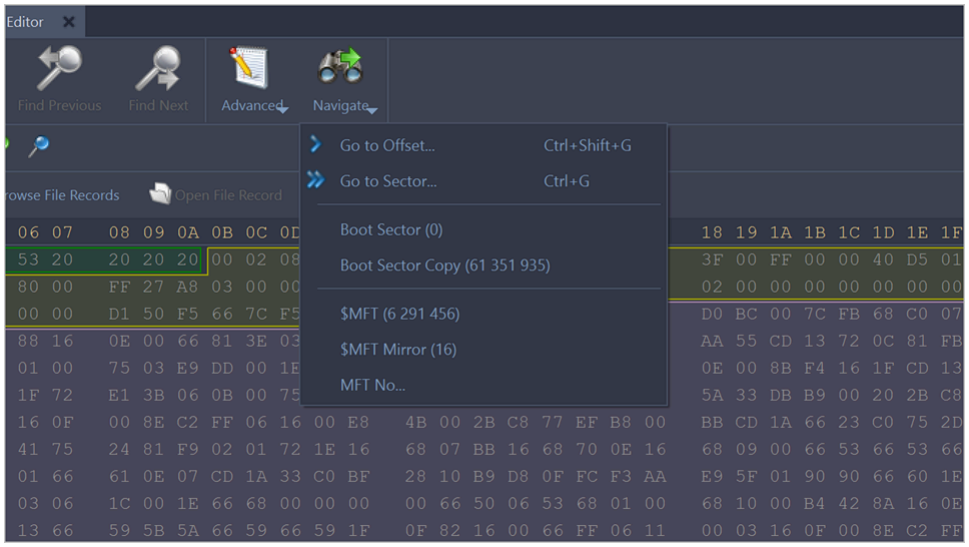


Figure 40: Example. Navigate Menu Selections

Related information

Navigation and Information on page 90
Move to Offset on page 94

Move to Sector (cluster) on page 94
Navigate a Physical Disk on page 92

Move to Offset

The ***Go to Offset*** menu opens the Go to Offset dialog allowing specification of an exact location (offset) in the disk to jump to.

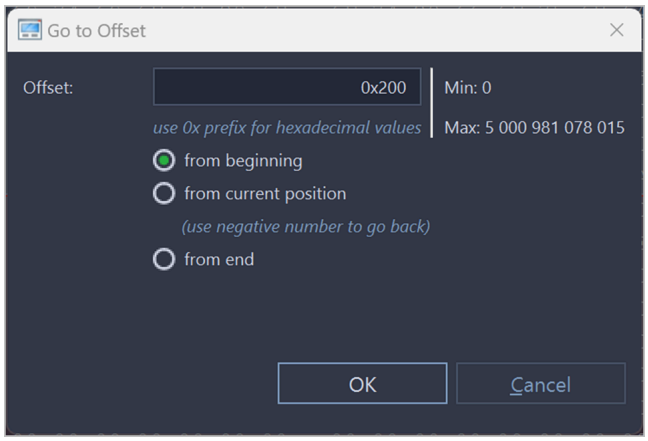


Figure 41: Go to Offset dialog

You can use both decimal and hexadecimal values, preceding hexadecimal values with 0x. For example, to specify location 512 as a hexadecimal number, enter 0x200. There are also options to specify an offset from the beginning, from the current position, or from the end.

Next to the offset edit field there are two labels specifying the minimum and maximum allowed vales for offsets displayed as decimal numbers.

You can also open this dialog directly by using the shortcut ***CTRL+Shift+G***.

Related information

Navigation and Information on page 90
Move to Sector (cluster) on page 94
Navigate a Physical Disk on page 92
Navigate a Logical Drive on page 93

Move to Sector (cluster)

This command allows jumping to the beginning of a specified sector or cluster.

There are two edit fields in this dialog that allow entering a desired location either as a sector number or a cluster number.

> ⚠ **Attention:** The ***Cluster edit field*** is available only for logical disks (*Volume*) and grayed out for all other objects.

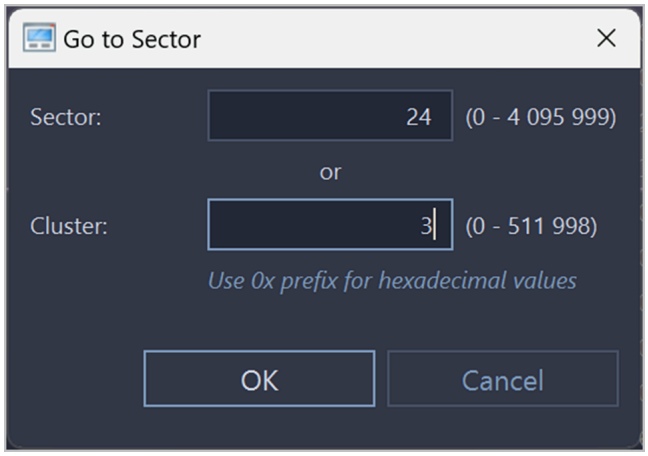


Figure 42: Go to Sector dialog

As with the offset dialog, you can also use both decimal and hexadecimal numbers.

Next to the edit field is the range of allowed values in brackets. Notice that not all sectors correspond to clusters, but every cluster corresponds to a particular sector.

You can enter either a sector value or a cluster value. Depending on which field is active, the dialog will use a sector or cluster. If you enter a number in the cluster edit field, a corresponding sector is displayed automatically.

You can also open this dialog directly using the shortcut ***Ctrl+G***.

Related information

Navigation and Information on page 90
Move to Offset on page 94
Navigate a Physical Disk on page 92
Navigate a Logical Drive on page 93
Local Devices on page 23

Move to MFT Record

Use the ***Navigate*** button in the ***Command bar*** to navigate to the disk system records of a *Logical drive*.

For *NTFS* volumes, the navigation menu will have following access points:

Boot Sector
Use this to navigate to the Boot Sector.

Boot Sector Copy
Use this to navigate to the Boot Sector Copy.

$MFT or MFT records (Master File Table)
Use this to navigate to the Master File Table file.

$MFT Mirror
Use this to navigate to the mirror of the MFT file.

***File record* №...**
Use this to navigate to an arbitrary MFT record. You can enter an arbitrary or a required record number in the Go to MFT Record dialog. To do this, click on the ***Navigate*** > ***MFT №...*** button in the Command bar.

The access points, when used, automatically select a corresponding template. For example, if a boot sector access point is selected, a boot sector template is applied to the boot sector offset.

Related information

Navigation and Information on page 90
Navigate a Physical Disk on page 92
Navigate a Logical Drive on page 93
Move to Offset on page 94
Move to Sector (cluster) on page 94
Move to Catalog Node on page 96

Move to Catalog Node

Use the ***Navigate*** button in the ***Command bar*** to navigate to the disk system records of a *Logical drive*.

For *file systems* as *HFS+* volumes, the navigation menu will have following access points:

Volume Header
Use this to navigate to the Volume Header.

Volume Header Copy
Use this to navigate to the Volume Header Copy.

Catalog File
Use this to navigate to the Catalog File.

***Node* №...**
Use this to navigate to an arbitrary Node record. You can enter an arbitrary or a required Node number in the Go to Catalog Node dialog. To do this, click on the ***Navigate*** > ***Node №...*** button in the Command bar.

The access points, when used, automatically select a corresponding template.

Related information

Navigation and Information on page 90
Navigate a Physical Disk on page 92
Navigate a Logical Drive on page 93
Move to Offset on page 94
Move to Sector (cluster) on page 94
Move to MFT Record on page 95

Search in Disk Viewer

To search text or byte sequence in Disk Viewer view :

- Click ***Ctrl+F*** shortcut key or
- Use ***Find...*** button in Disk Viewer's toolbar then Find text dialog will appear or
- Press ***Find...*** button in menu Edit.

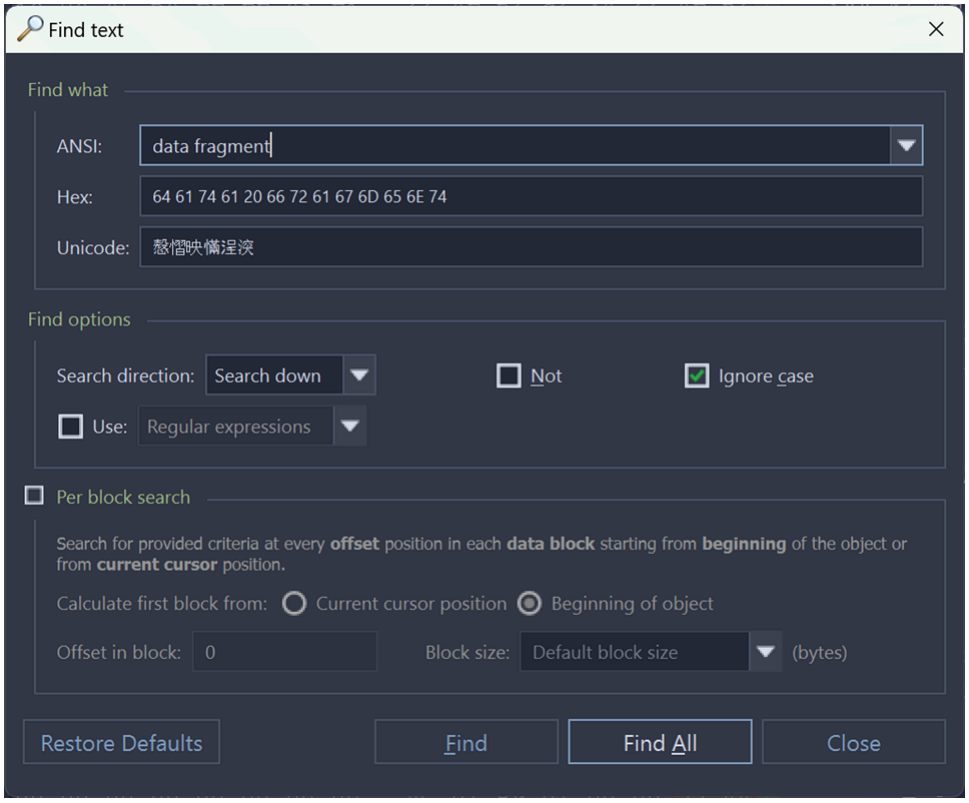


Figure 43: Dialog Find

Dialog options

Find what

Search pattern to find. Required. Can be set in one of the following formats:

- *ANSI* - text pattern, Regular expressions and wildcards can be used. History of ANSI search patters is preserved for next sessions and can be selected from drop-down list.
- *HEX* - search pattern in **hexadecimal** format.
- *Unicode* - search pattern in **Unicode** format.

> **Note:** When search pattern is entered in one of the find what text fields, the other two related fields will interpret entered value in correspondent format.

Find options

Regular expressions on page 102 and Wildcards on page 102 can provide even greater search capabilities.

Search direction will specify search direction from the current cursor position.

Not option search for characters that do not correspond to the ***Find what*** parameter.

Ignore case disables case-sensitivity in text search

Per block search

When this option is on, then search applies on per block fragments of context object. This method could be useful to search for repeated pattern, for instance at certain position (offset) in each and every sector (block).

Find command will initiate search process and will pause at first search result entry. Use ***Next*** button on dialog or ***F3*** keyboard shortcut to continue paused search.

When using ***Find All*** command, list of all search entries will appear in Find Results view. Use this list to navigate between search result entries (if any).

Find INSI data

Searching UU *ANSI* data with the follow parameters: ignore case, per block search with calculate first block from current cursor position, offset in block 3 and 1024 bytes block size.

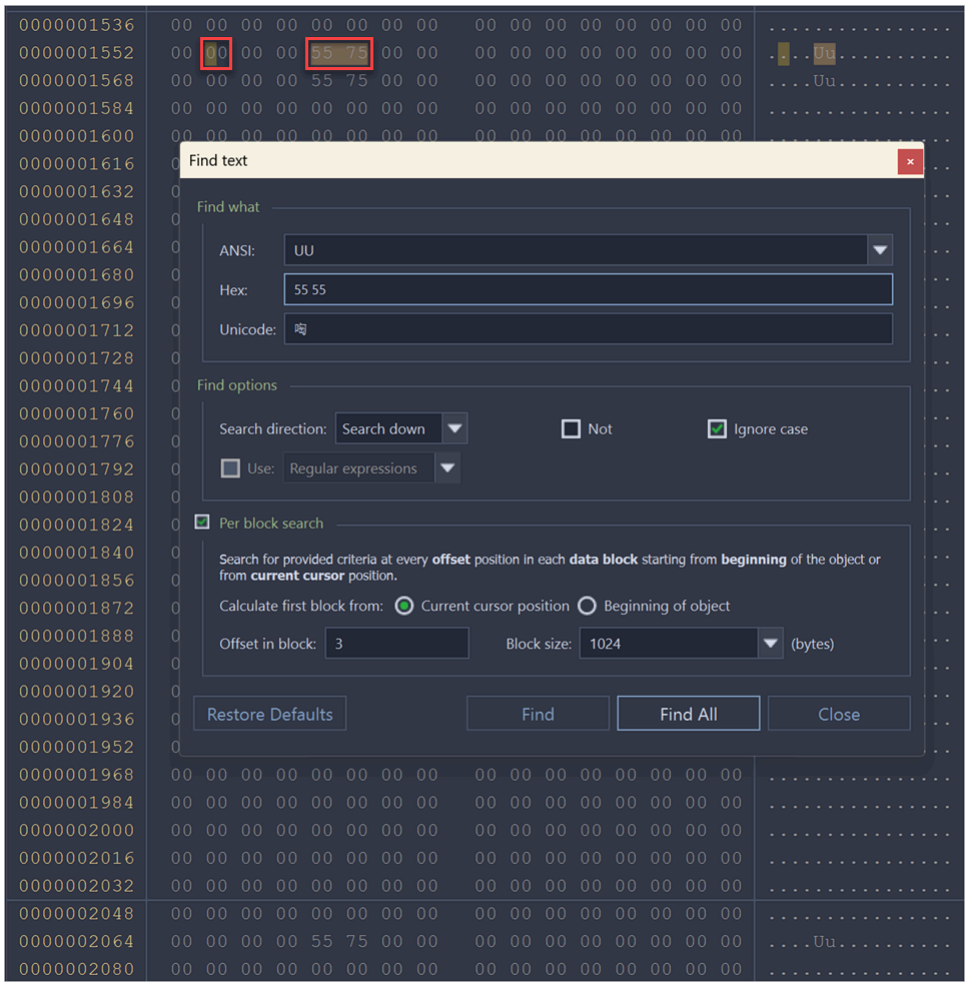


Figure 44: Example of data search

Note: Do not change the cursor position during per block search with calculate first block from current cursor position

Using the option ***Ignore case*** allowed to expand the search to the next variants: UU, uu, Uu, uU.

Current cursor position parameter allows to specify the beginning of the block, the block size is specified in the field ***Block size***.

Thus, the value located at the *Offset* 1556 was found, but the value at the offset 2068 was not found.

Find data using regular expressions

Searching ANSI data with the follow parameters: use regular expression U{2}[0-9], ignore case.

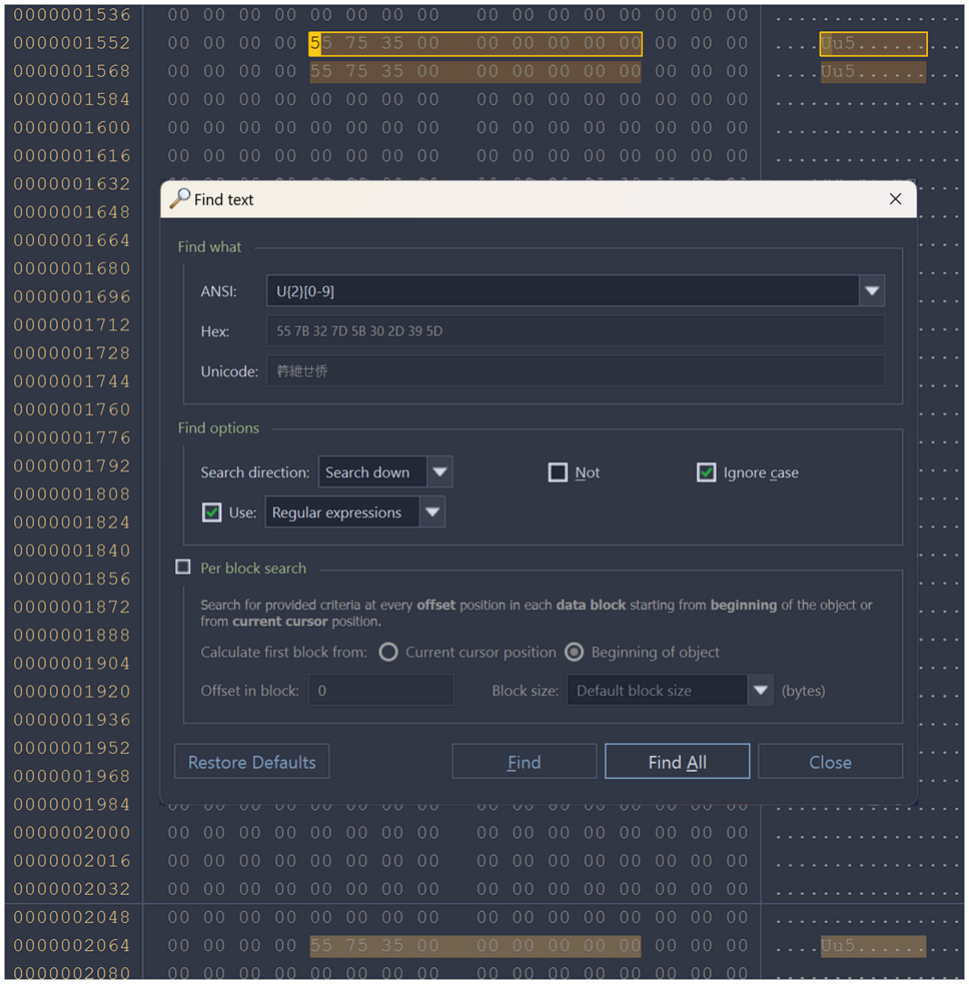


Figure 45: Example of data search using regular expressions

Using ***Regular expression*** U{2}[0-9] allow to search two symbols U successively and digit after they.

Browsing search results

After search completed, result entries (if any) are listed in Find Results view, grouped in subsequent search results, with offset (address) and short preview snippet. To focus on individual search result entry double-click on it in list or use **Goto** button in view's toolbar.

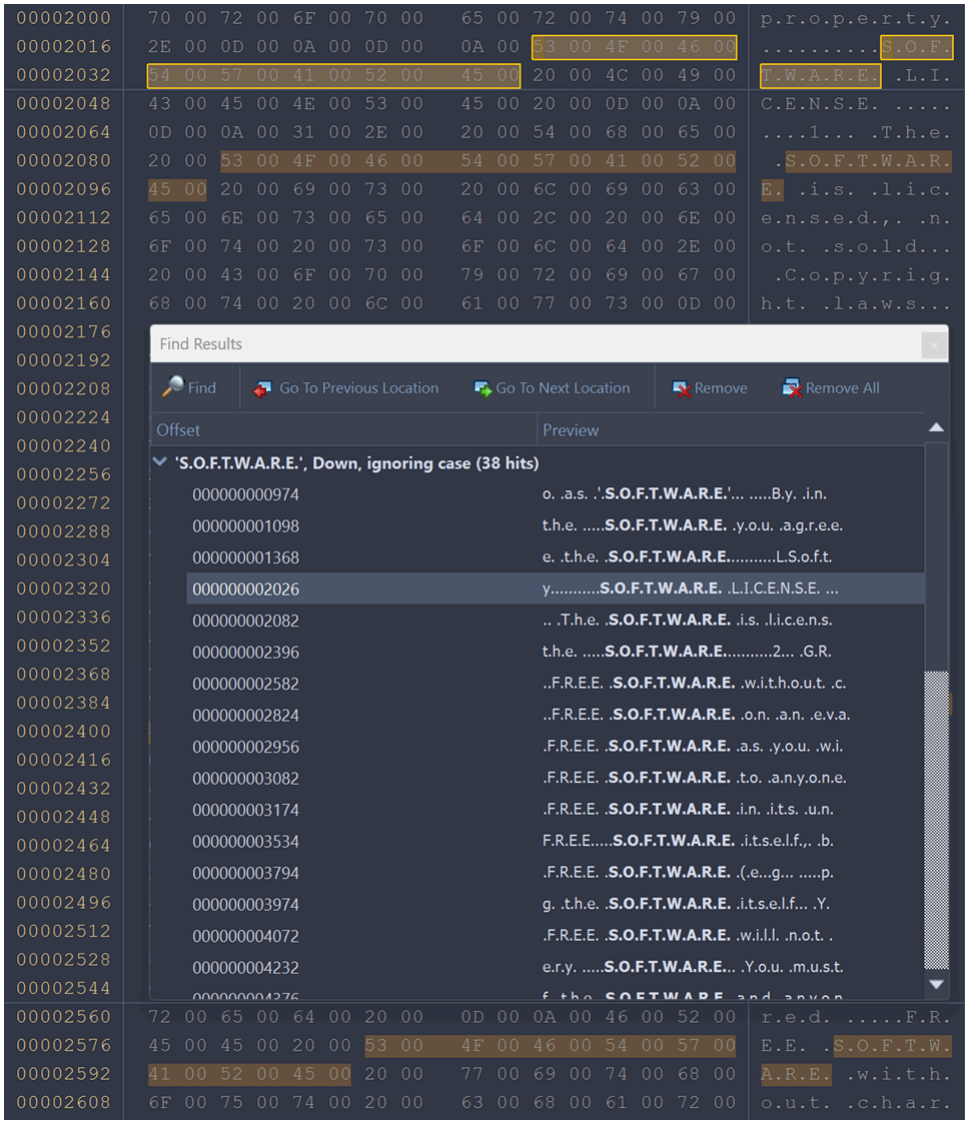


Figure 46: Find results in Disk Viewer

All search results are highlighted in Disk Viewer view.

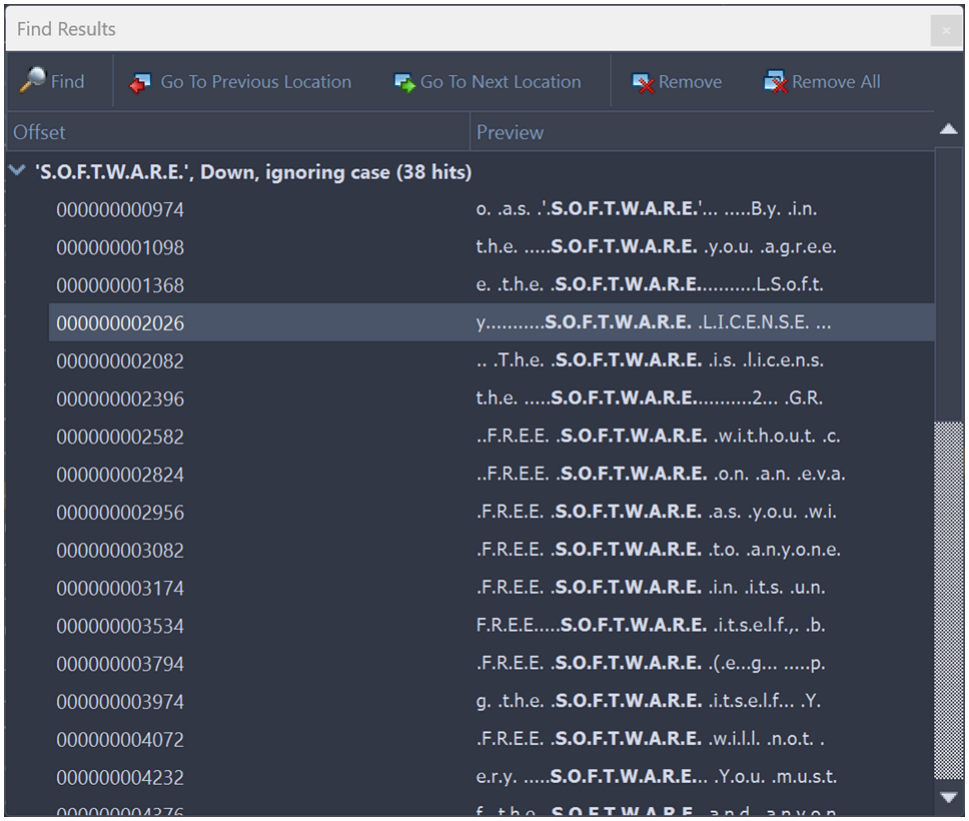


Figure 47: Find Results view

View actions

Find

Open Find dialog for a new search

Go Previous Location

Move cursor to previous search result

Go To Next Location

Move cursor to next search result

Remove

Remove selected search result or search group from list

Remove All

Clear search result list

Use context menu in Find result view to interact with each search entry individually.

Related information

Advanced tools and views on page 109
Data Inspector on page 109
Active bookmarks on page 112

Searching patterns on page 102

Searching patterns

Wildcards

A *wildcard* is a character that can be used as a substitute for any of a class of characters in a search. Wildcard characters are often used in place of one or more characters when you do not know what the real character is or you do not want to enter the entire name. In Disk Viewer three types of wildcard are used: **star** or asterisk(*), **question mark** (?) and **number sign** (#).

Examples of using wildcards:

Wildcard character	Example	Description
Asterisk (*)	docum*	Use the asterisk as a substitute for zero or more characters if you are looking for a file that you know what it starts with and you cannot remember the rest of the file name. The example locates all files of any file type that begin with **"docum"** including *documents.txt*, *document_01.doc* and *documentum.doc*.
	docum*.doc	To narrow the search to a specific type of file, include the file extension. The example locates all files that begin with **"docum"** and have the file name extension **.doc**, such as *document_01.doc* and *documentum.doc*.
Question mark (?)	doc?.doc	Use the question mark as a substitute for a single character in a file name. In the example, you will locate the file *docs.doc* or *doc1.doc* but not *documents.doc*.
Number sign (#)	doc_###.doc	Use the number sign (also known as the pound or hash sign) as a substitute for a single number in a name. In the example, you will locate the file *doc_012.doc* or *doc_211.doc* but not *doc_ABS.doc*.

Regular expressions

Regular expressions are special search patterns, more capable then wildcards to define search criteria.

Examples of using regular expressions:

^\d\d?$ - match integers 0 to 99

^\S+$ - match strings without white space

\b(mail|letter|correspondence)\b - match strings containing 'mail' or 'letter' or 'correspondence' but only match whole words i. e. not 'email'

&(?!amp;) - match ampersands but not &

\b(Eric|Eirik)\b - match Eric or Eirik

Using Templates

You can view system records (like *Boot Sector*, *Master Boot Record (MBR)*, *MFT or MFT records (Master File Table)* etc.) by using a *Templates (Patterns)* tool window. Template window is a small dockable window normally located to the left from main Disk Viewer editing area. If it is not visible, you can turn it on by selecting toolbar menu ***View*** > ***Windows*** > ***Templates*** .

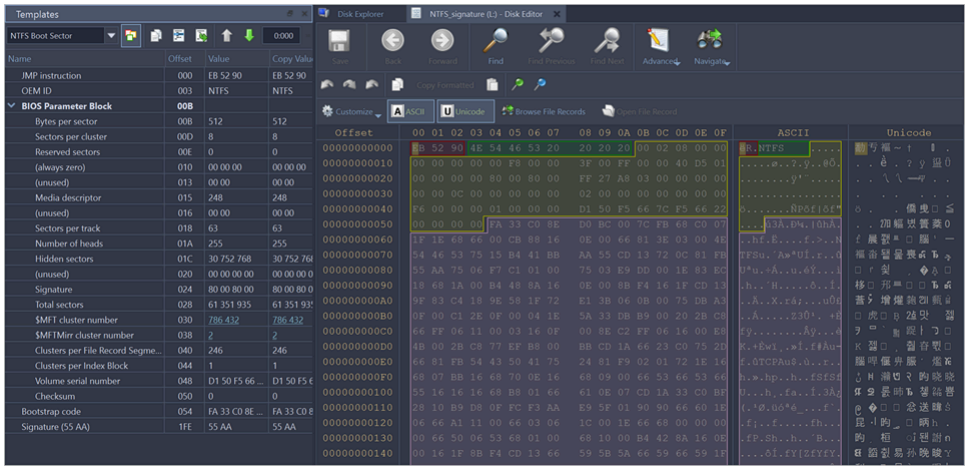


Figure 48: NTFS Boot Sector and its template

Applying a template

In order to apply a template to the desired offset, move the cursor to the location and use context menu command ***Set Template position***. The next step select a required template from the list box with template names in the toolbar of templates window.

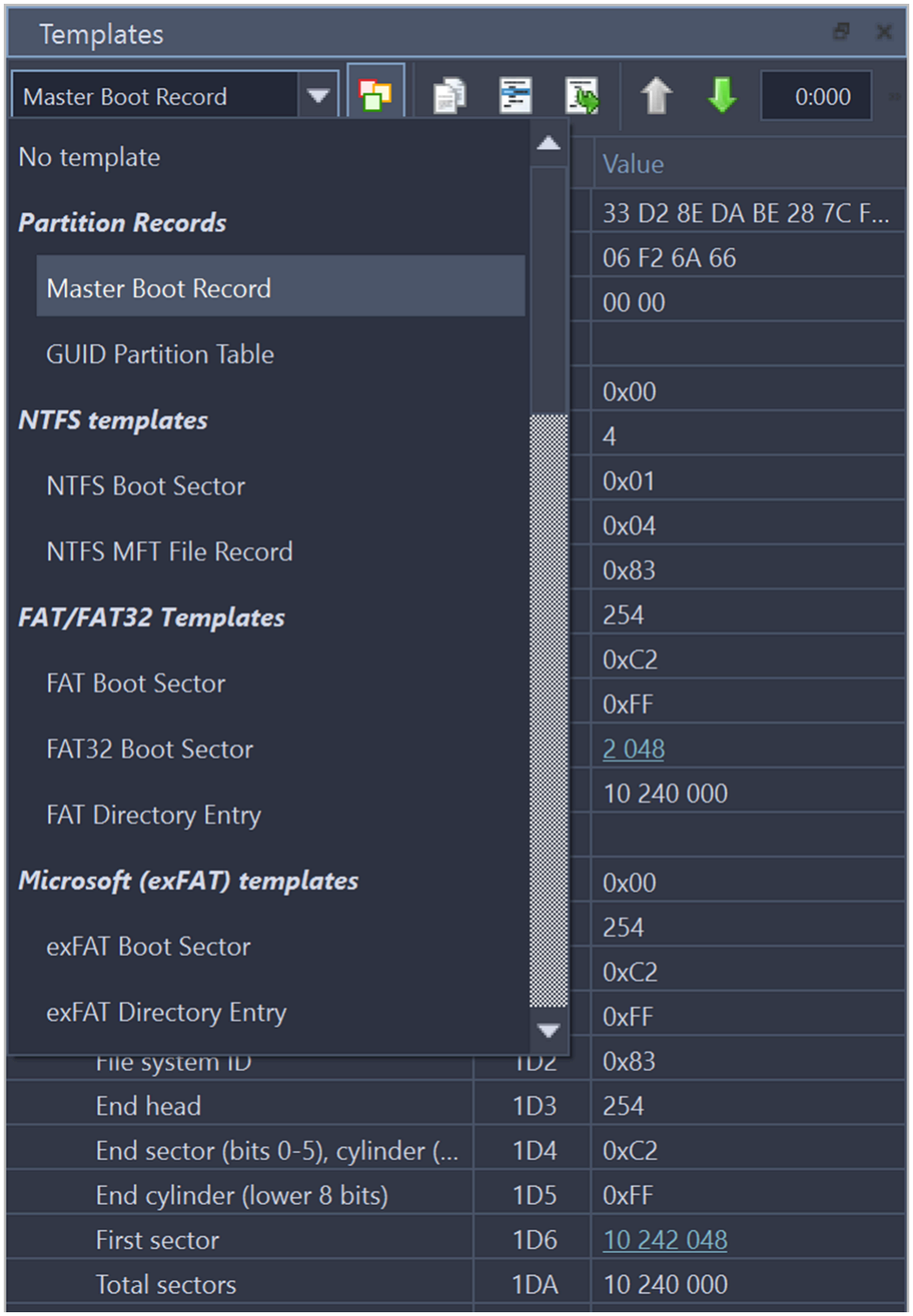


Figure 49: Templates list

When you are jumping to particular system areas using ***Navigate*** menu, the corresponding template might be applied automatically. This is true for templates like boot sectors, MBR or MFT record but not all access points have a template associated with them.

The following templates are supported:

Partition records

- *Master Boot Record (MBR)*
- *GUID Partition Table (GPT)*

NTFS **templates**

- NTFS *Boot Sector*
- NTFS *MFT File record*

FAT File System on page 142 templates

- FAT Boot Sector
- *FAT32* Boot Sector
- *FAT (File Allocation Table) Directory Entry*

exFAT **templates**

- exFAT Boot Sector
- exFAT Directory Entry

Microsoft *ReFS* templates

- ReFS Boot Sector
- ReFS Entry

Apple Hierarchical File System (*HFS+*) templates

- HFS+ *Volume Header*
- HFS+ Catalog *Node*
- HFS+ File Record

Apple (*ApFS*) templates

- ApFS Volume *Superblock*
- ApFS File System
- ApFS *Inode*

Linux Extended File System (*Ext*) templates

- *Ext2*/*Ext3*/*Ext4* Superblock
- Ext2/Ext3/Ext4 Inode

Linux B-tree (*BtrFS*) File System templates

- BtrFS Superblock

Linux/Unix File System (*JFS*) templates

- JFS Volume Superblock
- JFS Inode

Linux/Unix File System (*XFS*) templates

- XFS Volume Superblock
- XFS AG Free Space Block
- XFS AG Inode Block
- XFS AG Inode

Unix File System (*UFS*) templates

- UFS Superblock
- UFS Inode

Logical Disk Manager (*LDM*) templates

- LDM Private Header
- LDM TOC Block
- LDM VMDB Block
- LDM Klog Block
- LDM VBLK Block

Template Copy

The following templates have their copy:

- NTFS Boot Sector
- FAT32 Boot Sector
- HFS+ Volume Header
- Ext2/Ext3/Ext4 Superblock
- LDM Private Header
- LDM TOC Block

In this case template window will have an additional column named *Copy Value* which contains the data from the copy record. Template copies are useful to compare record located in different locations using the same pattern, for example to compare a boot record with its copy.

In case of *Copy* template its location is set separately from a main record using the same pattern. If the main template and its copy are intersecting, the copy template data will be shown in template window but not highlighted in the main edit area.

Setting template position

In order to set a template position or change an existing one move the cursor to desired location and use context menu command ***Set Template position*** (or ***Set Template Copy Position*** for its copy).

Navigating to a system area which has an attached template using ***Navigate*** menu also changes template position.

In order to facilitate the movement between records located in sequence, use arrow buttons located in the template window toolbar next to the templates list. For example, if you are editing or viewing an MFT record you can easily move to the next or previous record using those buttons.

Another way to set a template position is to enter new offset directly into template offset edit field in the template window toolbar. One of those fields are used for entering an offset of the main record and another is for its copy. The format of offset used in offset field is <sector:>:<sector offset>. You don't need to specify sector offset if you want to move to the beginning of the sector. For example, you can simply enter 100 to go to sector 100 and template offset will be shown as 100:0, but if you need to specify 128 byte in sector 100, you have to enter 100:128.

Highlighting template fields

By default all individual fields of template record are highlighted in Disk Viewer main area (in hexadecimal and ASCII columns only). This coloring highlighting can be disabled by clicking Toggle template fields coloring button in template window toolbar next to arrow buttons.

The colors used by template coloring are arbitrary and have no specific meaning, their main purpose is to make separate fields visible and distinguish from each other. Actually, a palette of several colors is chosen and colors are used in a circle. When you select a field in the template window, the current field is also highlighted in hex editing area with bold field frame.

When you move a mouse cursor above colored field in editing area, the name and value of the corresponding field is also shown in a tooltip.

Navigating around template fields

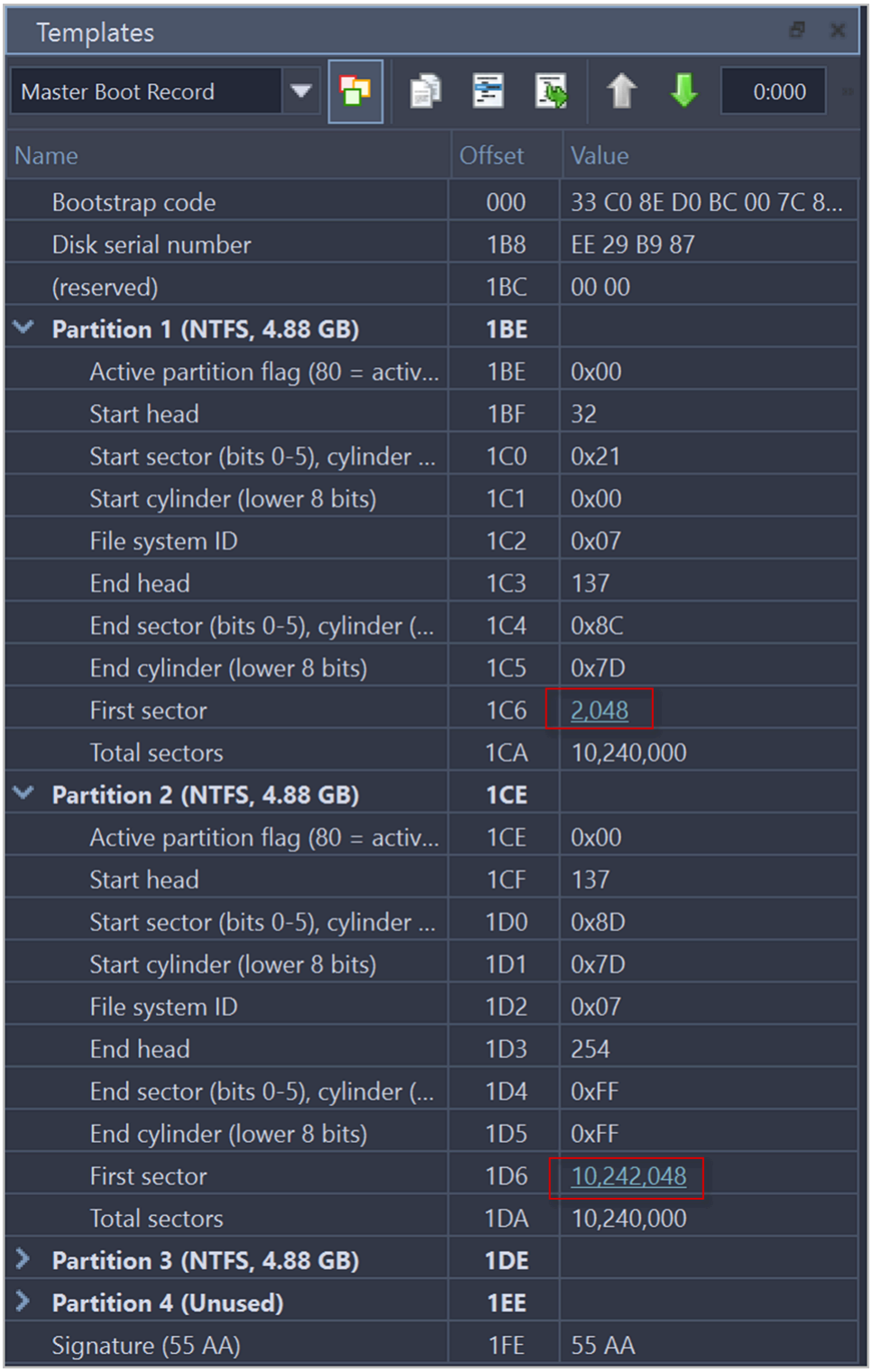
You can set the cursor (current position) to a particular field in a template by double clicking it. If you double click in *Name*, *Offset* or *Value* column, the position inside the main record is selected, but if you click inside *Copy Value* column, the navigation is performed to the field in template copy.

> ⚠ **Attention:**
>
> Please note, that in Edit mode double clicking inside of *Value* or *Copy Value* starts editing of the field instead of navigating to that field.

Hyperlinks in templates

Many templates contain hyperlinks allowing navigate easily to important data points.

For example, MFT records contain links to first cluster in data runs and MBR provides links to partitions.



Name	Offset	Value
Bootstrap code	000	33 C0 8E D0 BC 00 7C 8...
Disk serial number	1B8	EE 29 B9 87
(reserved)	1BC	00 00
Partition 1 (NTFS, 4.88 GB)	**1BE**	
Active partition flag (80 = activ...	1BE	0x00
Start head	1BF	32
Start sector (bits 0-5), cylinder ...	1C0	0x21
Start cylinder (lower 8 bits)	1C1	0x00
File system ID	1C2	0x07
End head	1C3	137
End sector (bits 0-5), cylinder (...	1C4	0x8C
End cylinder (lower 8 bits)	1C5	0x7D
First sector	1C6	2,048
Total sectors	1CA	10,240,000
Partition 2 (NTFS, 4.88 GB)	**1CE**	
Active partition flag (80 = activ...	1CE	0x00
Start head	1CF	137
Start sector (bits 0-5), cylinder ...	1D0	0x8D
Start cylinder (lower 8 bits)	1D1	0x7D
File system ID	1D2	0x07
End head	1D3	254
End sector (bits 0-5), cylinder (...	1D4	0xFF
End cylinder (lower 8 bits)	1D5	0xFF
First sector	1D6	10,242,048
Total sectors	1DA	10,240,000
Partition 3 (NTFS, 4.88 GB)	**1DE**	
Partition 4 (Unused)	**1EE**	
Signature (55 AA)	1FE	55 AA

Figure 50: Hyperlinks in template

Advanced tools and views

Disk Viewer delivers several tools for advanced users:

Data Inspector on page 109
Tool-view interpret currently selected data to several most used data types.

Active bookmarks on page 112
Provides ability to mark certain locations on edited subject to faster access and navigation.

Search in Disk Viewer on page 96
Enhanced search within edited content.

Related information

Property View on page 116

Data Inspector

The Data Inspector is a small viewing tool that provides the service of "inspecting" (or interpreting) data currently selected in the edit pane. The Data Inspector lets you view the type of data you have selected. This can help you interpret data as displayed in Disk Viewer.

To open the Data Inspector from the main menu choose ***View*** > ***Windows*** > ***Data Inspector***.

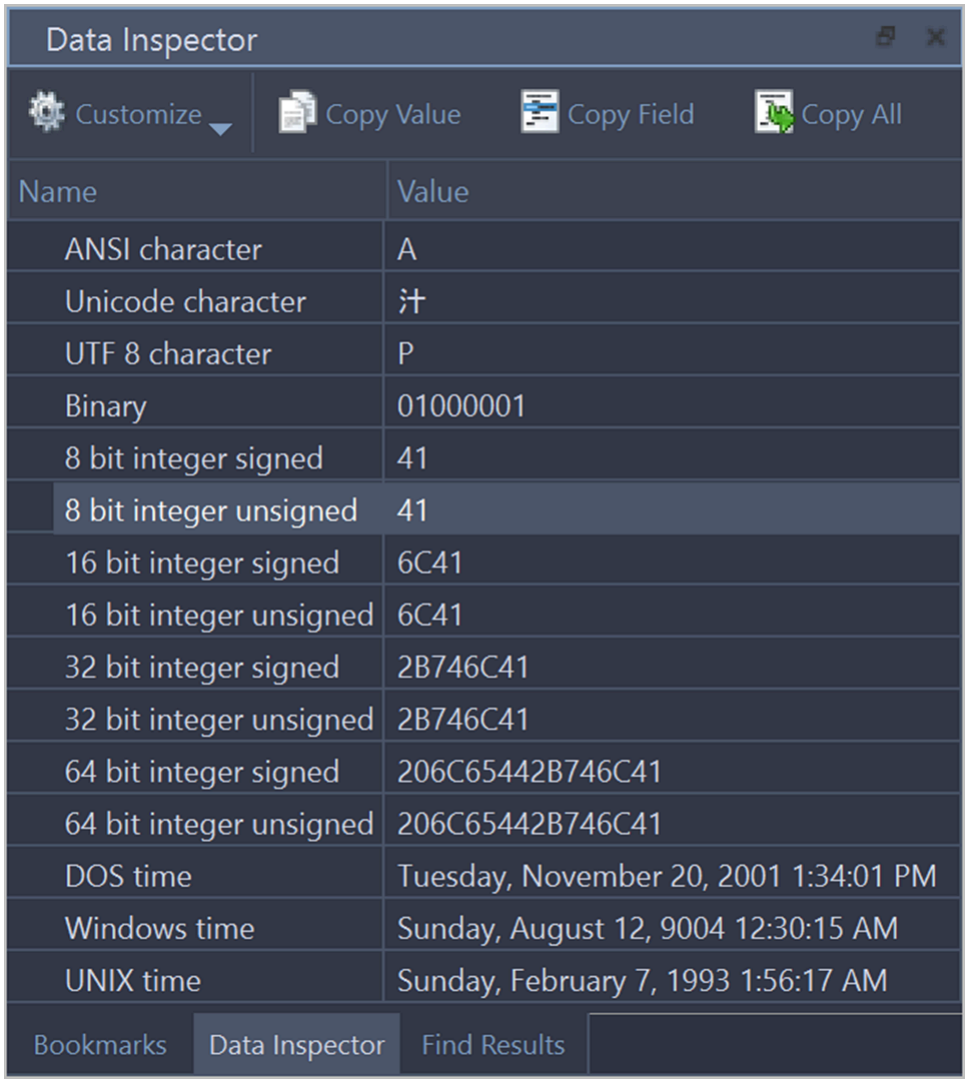


Figure 51: Data Inspector

View options:

Customize

These settings let you customize appearance for better experience with data.

Copy Value

Copy value of selected field to clipboard.

Copy Field

Copy entire selected field (value and field name) to clipboard.

Copy All

Copy all name and value fields in a view to clipboard.

Use view context menu to execute these commands for selected item (field).

Customize options:

Interpretation fields

Turn on/off interpretation fields, such as: ANSI, Unicode and UTF 8 characters, Binary, 8, 16, 32, 64 bit integer signed and unsigned, DOS, Windows and Unix times. Show All option turns on all fields and Show Service Fields option displays fields such as Raw (HEX) value, Sector, Offset.

Fields formatting

Interpretation filed could show data in one of the following formats:

Decimal
Changes data basis to decimal;

Hexadecimal
Changes data basis to hexadecimal;

Octal
Changes data basis to octal.

Use locale

Displays time/data in local format.

Big Endian

Toggle between *little endian* and *big endian* value representation. See: *Big Endian/Little Endian*.

The Data Inspector window is dockable and its location can be changed by clicking on the window title and dragging it to the new one. If the Data Inspector window is sharing its space with other tool views, you can change its relative position by left clicking and dragging the window tab. You can close the window by clicking on the [***X***] button in the top right corner of the window and reopen it again using the ***View*** menu in the main menu.

Related information

Advanced tools and views on page 109
Active bookmarks on page 112
Search in Disk Viewer on page 96

File cluster chain

File cluster chain is one of the essential approach to analyze file data integrity and file recovery. To help navigate through the content of an open file, file's cluster chain, shown sequence number, offset and size of each chain, is displayed in File Cluster Chain view.

To open the File Cluster Chain View form main menu choose ***View*** > ***Window*** > ***File Cluster Chain***

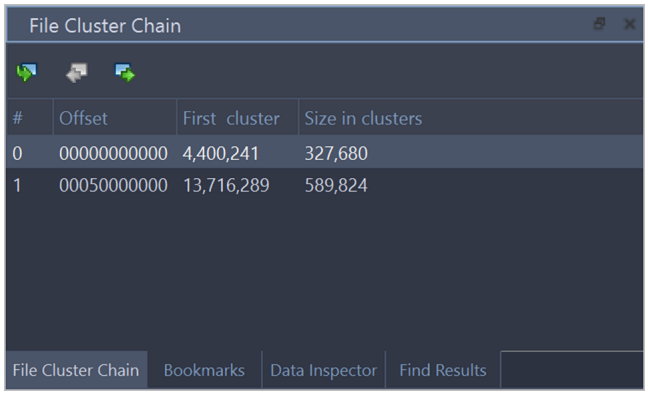


Figure 52: File Cluster Chain

View options

Go to

Go to selected cluster of cluster chain. Same effect can by achieved by double-clicking on cluster entry in cluster chain list.

Go to Previous

Go to previous cluster chain in sequence

Go to Next

Go to next cluster chain in sequence.

Related information

Advanced tools and views on page 109

Active bookmarks

Bookmarks allow you to save the current cursor location and quickly return to it later on. You may also give a name to a bookmark to make orientation easier.

Bookmarks are shown in the tool window called Bookmarks. If the Bookmarks window is closed you can open it using the main menu ***View*** > ***Windows*** > ***Bookmarks***.

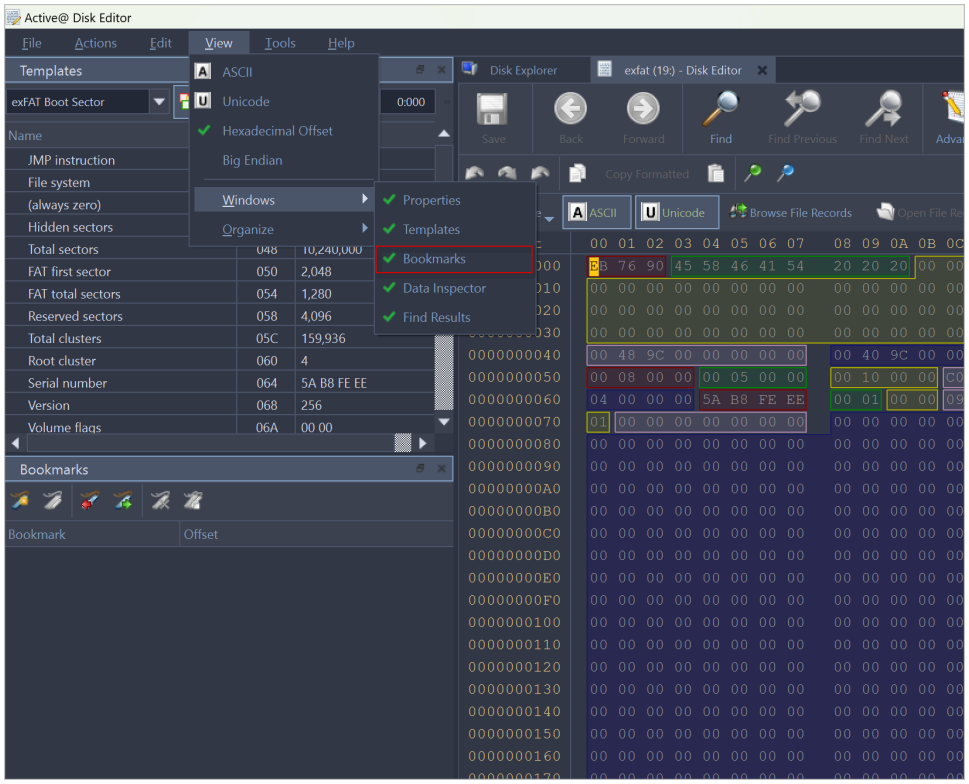


Figure 53: Open bookmarks view

Bookmark view

All bookmark for currently edited object are listed in Bookmark view. Bookmark will be saved for next session use if edited object is saved or left open before application exit.

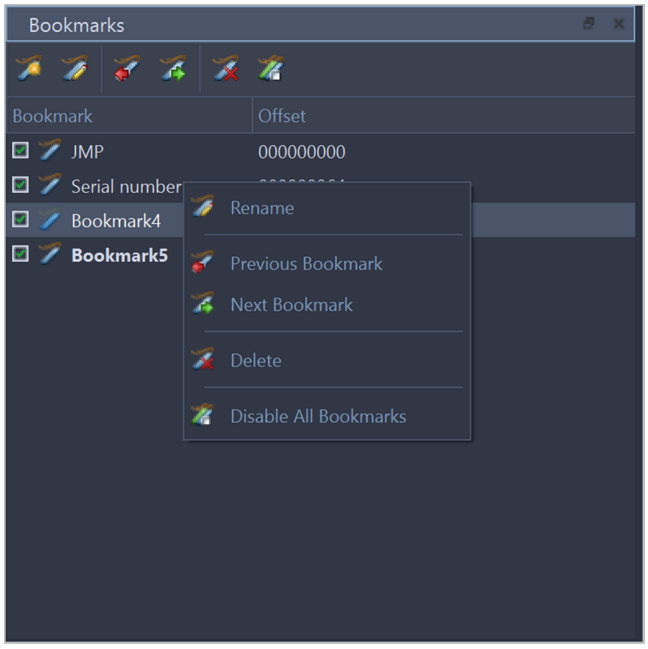


Figure 54: Bookmarks and Bookmarks view

View options

Toggle bookmark

Add or remove bookmark at current cursor position

Rename

Rename selected bookmark

Go to previous

Move (jump) cursor to previous bookmark

Go to next

Move (jump) cursor to next bookmark

Delete

Delete selected bookmark

Disable all bookmarks

Disable bookmarks, thus they will be ignored in bookmark's shortcut navigation.

Tip: Use context menu for selected bookmark for the same set of commands as in view's toolbar.

Add and remove a bookmark

Move cursor to position of interest in the Disk Viewer and press ***Ctrl+F6*** in order to add a bookmark or toggle a ***Toggle a Bookmark*** button in the Bookmark view toolbar. Alternatively, you can right click in the Disk Viewer and select a command ***Bookmarks*** > ***Toggle Bookmark*** from a context menu. The bookmark position is shown with a light blue box in the Disk Viewer and also added to the list of bookmarks in the Bookmark view.

To remove a bookmark, press ***Ctrl+F6*** while having the cursor over the position of that bookmark. You can also remove a bookmark from the Bookmarks view by selecting a bookmark in the list and clicking ***Delete*** button in a toolbar. The delete function can also be selected from a context menu and Bookmark view toolbar.

Going to a bookmark

If you have defined bookmarks, pressing ***F6*** will move your current position to the next enabled bookmark in the list.

You can also right click a bookmark and select the ***Next*** bookmark command from a context menu, . Another option is to double click a bookmark name in the Bookmarks window.

Pressing ***Shift+F6*** will move your current position to the previous enabled bookmark in the list or you can also use the ***Previous*** bookmark command from a context menu.

Commands ***Next*** and ***Previous*** also present in the Bookmark view toolbar and context menu ***Bookmarks*** > ***Next (Previous)*** in the Disk Viewer view.

Edit bookmarks

Bookmarks are named automatically when they are placed. You can rename a bookmark in the Bookmarks window to give it some meaningful name. To do so make a single mouse click on the bookmark name and edit it. Press ***Enter*** to accept your changes or ***Esc*** to cancel editing and revert to the original name. You can also rename a bookmark by right-clicking on it and selecting the ***Rename*** command from a context menu.

You can use also all of these commands from a toolbar in the Bookmark view.

All bookmarks are highlighted in Disk Viewer for easy navigation.

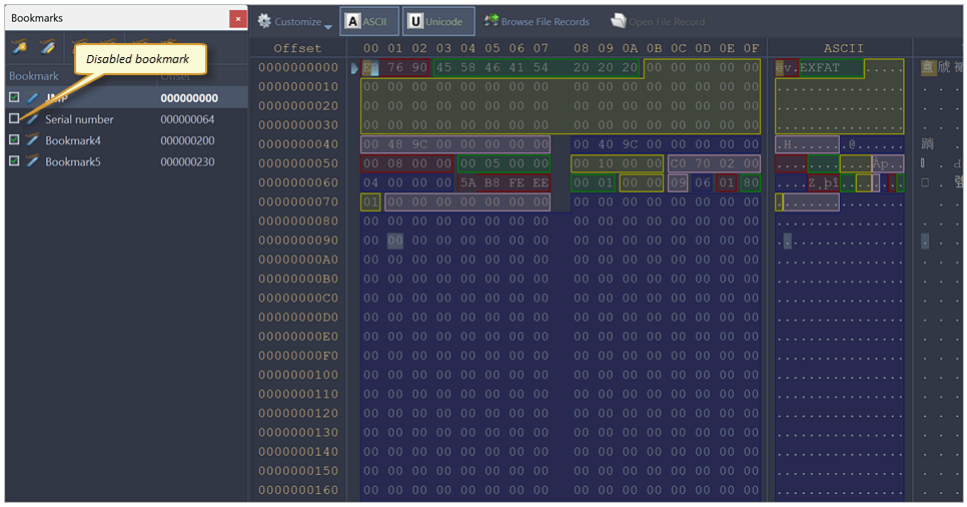


Figure 55: Bookmarks in Disk Viewer

Sometimes instead of deleting a bookmark it is useful to temporarily disable it. A disabled bookmark will not be counted when moving to the next bookmark. Uncheck a bookmark in the Bookmarks window to disable it. To disable all bookmarks at once click ***Disable*** all bookmarks in a toolbar or select this command in a context menu.

Related information

Advanced tools and views on page 109
Data Inspector on page 109
Search in Disk Viewer on page 96

Featured views

Active@ KillDisk offers a number of advanced views to work in conjunction with the software to make operations easier to perform and make disks easier to explore.

Application makes it possible to explore disks both on a file level (in File Browser) and on a low level (in Hexadecimal Viewer). Disk health analysis can be performed with the S.M.A.R.T. monitor.

Property View

To show detailed information about any subject of an application (such as disk, partition, volume, file etc.) **Active@ KillDisk** uses information views. When displayed these views show information about the object being selected in the Disk Explorer. If selected object is changed, displayed information refreshes. Also *S.M.A.R.T.* parameters are display for the selected disk (if the device supports it).

SMART data can be used to detect problem disks as long as important disk information has been reflected such as Power-on Hours, Reallocated Sectors and Current Pending Sectors.

When Current Pending Sectors parameter differs from zero, this means the disk has bad sectors. It will cause problems in the future. Dispose these disks as soon as possible.

To open Property view for selected item do one of the following:

- Click ***View*** > ***Windows*** > ***Properties*** from the main menu
- Press ***F4*** (keyboard shortcut)
- Click ***Properties*** command from object's context menu

Properties

Name	Value	Description
Name	PhysicalDrive0	
Platform Name	\\.\PhysicalDrive0	
BIOS Name	80h	
Product Name	ST5000NM0084-1HT17X	
Product Revision	SN05	
Serial Number	Z4D3BD6Y	
Port	1-00-00-00	
Status	Ready	
Type	Fixed Disk	
Size	4.55 TB (5,000,981,078,016 bytes)	
Free Space	2.48 MB (2,604,032 bytes)	Total size of all unallocated areas on this Disk;
Unallocated Span	1.19 MB (1,252,864 bytes)	Size of largest continuous unallocated chunk presented on
Device Geometry		
Partition Style	GUID Partition Table	
Partitioning	GPT (Basic)	
Total Sectors	9,767,541,168	
Bytes per Sector	512	
Bytes Per Physical Sector	4,096	
Disk Hidden Areas		
Visible Disk Sectors	9,767,541,168	
HPA Hidden Sectors	0	
DCO Hidden Sectors	0	
SMART Parameters		
Device Model	ST5000NM0084-1HT17X	
Serial Number	Z4D3BD6Y	
Firmware Version	SN05	
Capacity	4.55 TB (5,000,981,078,016 bytes)	
ATA Version	10	
ATA Standard	Device does not report version	
SMART Support	1	Self-Monitoring, Analysis and Reporting Technology Suppo
Off-line Data Collection Status	130	
Self-test Execution Status	0	
Time Off-line Data Collection, sec	575	Total time to complete Off-Line data collection
Off-line Data Collection Capabilities	123	
SMART Capabilities	3	
Error Logging Capabilities	1	
Short Self-test Time, min	1	Short self-test routine recommended polling time
Extended Self-test Time, min	255	Extended self-test routine recommended polling time
Attributes		
[001] Read Error Rate	218242045	
[003] Spin-Up Time	0	
[004] Start/Stop Count	3810	
[005] Reallocated Sectors Count	0	
[007] Seek Error Rate	56692309	
[009] Power-On Hours Count	10508	
[010] Spin-up Retries	0	
[012] Power Cycle Count	2650	

Figure 56: Property View Example

Besides displaying a valuable data it also allows you to copy that information into a clipboard by using context menu commands.

Context menu commands:

Copy
Copy information of selected object in property tree to the clipboard

Copy Value
Copy Value of selected field to the clipboard (value only)

Copy Field
Copy formatted Name and Value pair to the clipboard

Copy All
Copy all information as formatted set of Name and Value pairs

Copied Information Example

```
Fixed Disk PhysicalDrive0
  Disk Attributes.....................Fixed, HPA DCO Ready, Wipeable, Erasable, SMART Ready, Security
 Supported, Security Frozen
  Device Key..........................disk-ST5000NM0084-1HT17X-SN05-Z4D3BD6Y
  Node Attributes.....................Disk, Ready, Populated
  Controller Port.....................1-00-00-00
  Name................................PhysicalDrive0
  Platform Name.......................\\.\PhysicalDrive0
  BIOS Name...........................80h
  Product Name........................ST5000NM0084-1HT17X
  Product Revision....................SN05
  Serial Number.......................Z4D3BD6Y
  Port................................1-00-00-00
  Status..............................Ready
  Type................................Fixed Disk
  Size................................4.55 TB (5,000,981,078,016 bytes)
  Free Space..........................2.48 MB (2,604,032 bytes)
  Unallocated Span....................1.19 MB (1,252,864 bytes)
  Device Geometry
    Partition Style...................GUID Partition Table
    Partitioning......................GPT (Basic)
    Total Sectors.....................9,767,541,168
    Bytes per Sector..................512
    Bytes Per Physical Sector.........4,096
  Disk Hidden Areas
    Visible Disk Sectors..............9,767,541,168
    HPA Hidden Sectors................0
    DCO Hidden Sectors................0
  SMART Parameters
    Device Model......................ST5000NM0084-1HT17X
    Serial Number.....................Z4D3BD6Y
    Firmware Version..................SN05
    Capacity..........................4.55 TB (5,000,981,078,016 bytes)
    ATA Version.......................10
...
```

If any device is not selected in Disk Explorer or My Computer is selected in Disk Explorer's My Computer view, software and hardware information is displayed in the Properties view.

Copied Hardware and Software Information Example

```
 My Computer
  Workstation ID.....................System Serial Number
  System Information
    Operating System..................Windows 11 24H2
    Platform..........................64-bit
    Administrator Rights..............Yes
    User Name.........................Alex
    Host Name.........................Win11x64-1
    Kernel Version....................10.0.26100 (winnt)
  Version Information
    License...........................User License
    Registered to.....................LSoft Development
    Kernel Version....................15.10.28
    Product Version...................8.0
    File Version......................8.0.35
  Hardware Information
    Processor.........................Intel(R) Core(TM) i7-7700 CPU @ 3.60GHz
    Processor Architecture............x86_64
    Logical Processors................8
    Memory............................22,9 ГБ
    Motherboard.......................ASUSTeK COMPUTER INC. PRIME B250-PRO
    Motherboard Serial................161290869600098
    BIOS Name.........................American Megatrends Inc. 1205
    BIOS Serial.......................System Serial Number
    Desktop...........................1920 x 1038
    Controllers
      IDE Controllers
        Intel(R) 300 Series Chipset Family SATA AHCI Controller
        SCSI Controllers
        LSI, SAS2 2008 Falcon
...
```

Related information

Disk Explorer on page 23

S.M.A.R.T Monitor view
Application Log on page 24

Application Log

Application Log reflects every action taken by the application and displays messages, notifications and other service information. Use these messages to observe and analyze processes flow.

To open Application log view do one of the following:

- Click ***Tools*** > ***Application Log*** from the main menu
- Press ***F8*** keyboard shortcut

Once Application Log View is open and active, you can use toolbar buttons and the context menu to perform the following tasks:

Save log as

Opens a standard ***Save As*** dialog. Save the actual application log file to the local disk Default is .LOG file extension.

Save hardware info as

Opens a standard ***Save As*** dialog. Save the disk diagnostic file to the local disk. Default is .XML file extension.

Log entry filter

Shows or hides specific entry types in Log View, such as System, Network or Security.

Text size

Changes text size from Small to Large. You can also use press and hold ***CTRL*** key and mouse wheel to adjust text to desired size.

Expand/Collapse All

Expands all collapsed log nodes.

Clear

Clear the log for the current application session.

Tip:

We recommend that you attach a copy of the log file to all requests made to our technical support group. The entries in this file will help us to resolve certain issues.

Output View

Simplified log view pane can be used as a convenient way to view application events of ongoing processes at any featured view. Context menu allows to filter out event entries by its type or significance (priority) attributes. This pane remains visible when switching between main featured views.

To open Output View do one of the following:

- Click ***View*** > ***Windows*** > ***Output*** from the main menu or
- Press ***CTRL + O*** keyboard shortcut

Related information

Disk Explorer on page 23
Property View on page 116

Advanced

Active@ KillDisk has a number of extra features to ensure the most complete sanitation operations, flexibility to meet the most strict requirements and compatibility with a wide range of systems. This section outlines these features:

Set Disk Serial Number

If you notice that disk serial number displayed in **Active@ KillDisk** does not match the number displayed on the label attached to the physical disk, **Active@ KillDisk** let you option to change it manually. To access this feature right-click the disk and select ***Set Serial Number*** from the context menu.

There are several methods of disk serial number detection, application pulls it from various sources: ***IOControl***, ***SMART*** and ***WMI*** (some of them can be disabled and grayed out, depending on Operating System support). Click the different options to apply different serial number detection method for the particular disk. Default serial number detection method applied to all disks can be set up in Preferences on page 72.

Note:

If you don't see your serial number in any of the detection methods try marking the ***Swap Symbols*** check box. If this doesn't help you can input disk serial number manually to be printed on certificates properly (ultimate option).

Reset Hidden Areas

Active@ KillDisk is able to perform erasing of a disk's hidden zones: *Host Protected Area (HPA)* and *Device Configuration Overlay (DCO)*.

To perform this task, right click on the disk and select ***Reset Hidden Areas*** from the context menu:

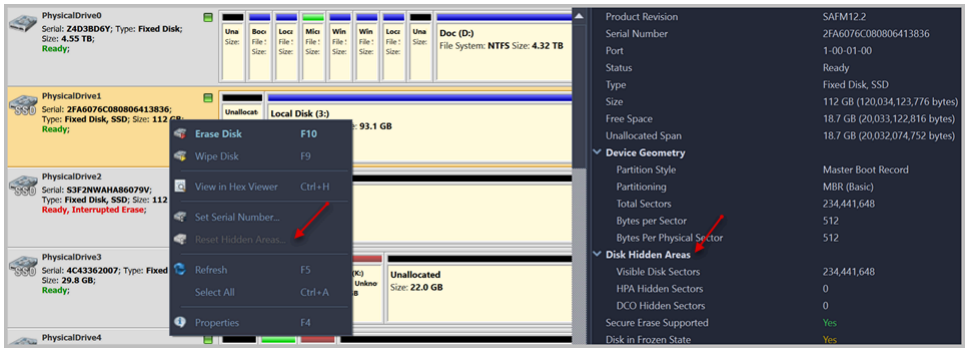


Figure 57: Disk Hidden Areas Reset

When related context menu item is disabled, this means that there are no hidden areas on the disk has been detected, so nothing to reset for the particular disk.

Related information

Disk Hidden Zones on page 202

Map Network Shares

This feature creates a specific local drive letter for remote locations to save logs and other documents to, as well as provides a central location for documents to be stored.

Note:

Map Network Shares is very useful feature when running application under Boot Disk and in Batch Mode.

To map a network share:

1. Open Mapping Dialog

   Navigate to ***File*** > ***Map Network Share...*** from the main menu.
2. Configure Mapping

   Assign a drive letter, type a network folder location or click ***Browse*** button to browse local network and select a proper network share. If sharing policy requires, type user name and password:

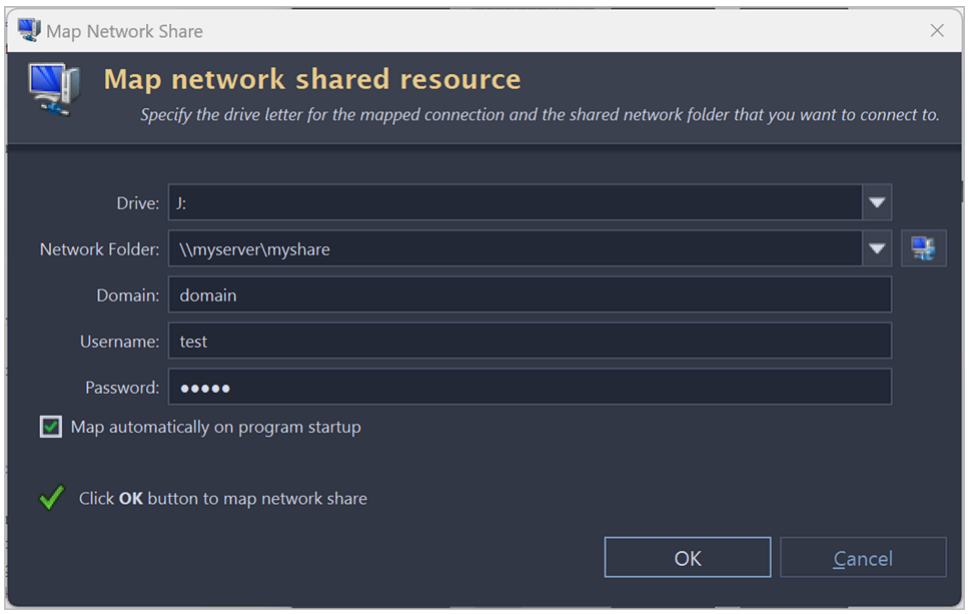


Figure 58: Mapping a Network Drive

If you want to save configured mapping for future use, make sure ***Map automatically on program start up*** is marked.

Note:

Active@ KillDisk will identify all connected network drives, so you may use the drop-down list to select the one you'd like to use

3. Attach a Network Share

   Click ***OK*** to attach a network share mapped to the local drive letter.

After your network drive is configured, you may select it as a destination for certificates and reports in Erase Certificate on page 77 or Processing Report on page 80.

Name Tags

Name Tags are used in various scenarios to create meaningful file names, label data, barcode data, and other outputs. Predefined constant values in brackets, such as *{SerialNumber}*, are replaced with the actual disk's serial number when a label or barcode is generated and printed. The following sections describe the available *Name Tags* grouped by category.

General

{Computer ID}
: Workstation (computer) ID

{OS}
: Operating System name

{AppName}
: Application name

{AppVersion}
: Application full version

{KernelVersion}
: Kernel version

{UniqueID}
: Generated unique 8 symbols ID

Tags to represent current date in different formats:

Date & Time

{Date(YYYYMMDD)}
: Complete date in full form without delimiters

{Date(YYYY-MM-DD)}
: Complete date in full form with delimiters

{Date(YYMMDD)}
: Complete date in short form without delimiters

{Date(YYYY)}
: Year in full form

{Date(YY)}
: Year in short form

{Date(Month)}
: Full month name as literal

{Date(MM)}

Month as digital with leading zero

{Date(DD)}

Day of month with leading zero

{Time(HHmmss)}

Time with hours, minutes and seconds without delimiters

{Time(HH-mm-ss)}

Time with hours, minutes and seconds with delimiters

{Time(HH)}

Hours with leading zero

{Time(mm)}

Minutes with leading zero

{Time(ss)}

Seconds with leading zero

Values for these name tags retrieved from the context device:

Disk Attributes

{Serial ID}

Disk serial number, retrieved from OS or from *S.M.A.R.T.* attributes

{Platform ID}

Disk platform identification (may be vary due to OS format)

{Product ID}

Disk manufacturer Id

{Model}

Disk model name (if available)

{Size}

Disk size in gigabytes

{Sectors}

Disk size in sectors

Disk processing attributes based on execution conditions:

Processing Attributes

{DiskCount}

Quantity of disk processed in group.

{Method}

Erase method

{Passes}

Erases passes description

{Verified}

Verification attribute

{DateStarted}

Process start date

{TimeStarted}

Process start time

{TimeElapsed}

Process elapsed time

{Status}

Overall completion status for group processing or separate disk processing status.

{StatusCode}

Overall process result digital code

Item processing attributes based on execution conditions:

Item processing attributes

{Sequence #} ... {Sequence 000#}

Sequential number. Used for group (batch) processing.

{ProcessType}

Process type name

{ProcessedAs}

Process short name

{Range}

Processed disk range

Disk Images

Disk Image definition and usage

Disk Image is a copy of your logical drive or physical device that is stored in one file. This can be useful when you want to backup the contents of the whole drive, and restore it or work with it later. Before you start recovering the deleted files, it may be a good idea to create a Disk Image for this drive, if you have enough space at another drive. Why? Because if you do something wrong while recovering the files (for example, recovering them onto the same drive could destroy their contents), you still will be able to recover these deleted files and folders from the Disk Image that you have wisely created.

When to use Disk Image

Raw disk images are very helpful in a data recovery. Here are some reasons why a raw disk image can be used for data recovery:

- Data recovery technologies are based on searching the unused space on a partition for traces of deleted, lost or damaged files and folders. So-called "unused space" on a partition is not recognized by the file system and is not saved to a regular disk image. However, this space does contain valuable data information and it is saved to a raw disk image.
- The uncompressed raw disk image file contains a sequence of sectors that is unchanged from the original. There are no headers or other application-specific identifiers added. As a result, the raw disk image can be viewed by any data rescue software as a mirror of your drive. If the integrity of the data on your live disk is questionable, you may want to experiment with the data on the partition image instead.
- If file size is an issue, a compressed raw image may be used. **Active@ UNDELETE** is an example of data recovery software which can work with both compressed and uncompressed raw images.
- Raw images have no regard for the file system type. During the raw disk image recording process, all sectors are backed up. An image of any partition can be restored by using **Active@ Disk Image** software.

- If you want the data from a file to be restored from the disk image to the same exact location as they were before, then use a raw disk image. A regular image saves all current data but restores files to different sectors, allowing the partition to shrink or grow, depending on the size of the replaced file. In a regular situation, you should not be concerned about partition size. If the partition size is important, however, a raw image is the solution.

Related information

Disk Explorer on page 23

Mount Disk Image

Mount *Disk Image* allows you to work with a specific disk image file as a virtual device. The application supports VMware virtual disks, VirtualPC disks, ISO image files, and any image in RAW format (sector-by-sector copy).

To mount (open) Disk Image:

1. Launch ***Mount Disk Image*** dialog in one of the following ways:
   - From the toolbar click ***Mount Disk Image...*** button;
   - Use command ***File*** > ***Mount Disk Image...*** from main menu;
2. Select Disk Image file name and other options

   Disk Image file name

   Full path to the location of disk image file

   Caption (Display name)

   Enter any label to distinguish newly opened (mounted) disk image among other devices and disks or use default (generated) Disk Image name.

   Auto-load mounted Disk Image when application starts

   Use this options to auto load mounted disk images each time when application starts.

3. Open Disk Image

   Click ***OK*** to confirm and open a Disk Image.

If a disk image has been mounted and opened successfully, the disk image node appears in Disk Explorer.

Related concepts

Disk Images on page 124
Disk Image definition and usage

Related information

Disk Explorer on page 23

Troubleshooting

In the events of technical difficulties with **Active@ KillDisk** you may choose to either troubleshoot the system yourself or, within an active maintenance period (you receive 1 year free with your purchase), you can contact our support team. Attach your Application Log on page 24 and hardware configuration (Hardware Diagnostic File on page 127) with your support request.

Related information

Common Tips on page 126
Application Log on page 24
Hardware Diagnostic File on page 127

Common Tips

Common Problems

Disk data can not be erased

Ensure that disk is fully functional (no physically damages) and is accessible by Operating System.

Ensure you are not erasing the system disk or the disk KillDisk launched from (application won't let you erase these disks).

Data still found after a Wipe operation

The Wipe operation sanitizes the only data that has already been deleted and not visible by Operating System. To sanitize ALL the data including existing files and the Operating System itself, use the ***Erase Disk*** operation.

Erased the wrong disk

Stop erase operation as soon as possible. Once the data completely sanitized, it won't longer be accessible. Use a tool like **Active@ File Recovery** (https://www.file-recovery.com) to recover remains of data that has not been sanitized yet.

Boot Disk Creator Problems

All the Operating Systems options are grayed out

Make sure you have KillDisk and Boot Disk Creator is registered and activated. You should see your registered name at bottom of the dialog.

Image file not found

Most likely you have activated the freeware package that does not have the Boot Disk image you wish to create (it does have the only tiny Console bootable image). Download a commercial package using the link provided after the purchase and reinstall the software.

Issues formatting USB drive

This may happen occasionally when the file system causes conflicts in Windows. Launch the **Active@ KillDisk** application and erase the first few megabytes of the USB drive you wish to use. Unplug and plug again USB disk, then try again. In most cases this solves the problem.

Issues booting from the boot disk

Make sure the boot disk device is set at the top of your boot priority in BIOS.

Make sure your system time in the BIOS is accurate.

Make sure you are not booting a 64-bit Boot Disk on an old 32-bit PC. In case of old hardware create and use a Console-based boot disk.

Hardware Diagnostic File

If you want to contact our technical support a file that contains a summary of your local devices and hardware configuration is very helpful and it is required to submit it for the proper problem investigation.

Application allows you to create a hardware summary file in XML format. This data format is "human-readable" and can help our technical support staff to analyze your computer configuration or point out disk failures or abnormal behavior.

To create a hardware diagnostic file, click ***Save Hardware Info as*** from the ***File*** menu .

⚠ **Attention:**

To save time on initial contact with our technical support staff we highly recommend that you submit a hardware diagnostic file, otherwise, most likely, it will be requested from you by our support team later on.

Related information

Common Tips on page 126
Application Log on page 24

Knowledge Base

This knowledge base provides essential information about data storage technologies and concepts that form the foundation of modern computing systems. Understanding these core principles is crucial for effective data management and troubleshooting.

HDD Organization
Physical and logical structure of hard disk drives, low-level disk organization

RAID Technology
Redundant Array of Independent Disks configurations for performance and data protection

LDM (Logical Disk Manager)
Windows dynamic disk management and volume organization

File System Organization
How operating systems structure and manage data on storage devices

Data Erasure Concepts
Methods and principles for securely removing data from storage media

Glossary
Comprehensive definitions of technical terms used throughout this guide

This knowledge base serves as both a learning resource and a quick reference for understanding how data is stored, protected, and managed in computer systems. Each section builds upon fundamental concepts to provide practical insights into real-world storage scenarios.

Types of storage media

Modern storage media can be divided into several main hardware types, each with its own characteristics, advantages, and typical use cases.

Hard disk drive(HDD) on page 210 is the oldest and most traditional type of storage still in widespread use. It consists of one or more spinning magnetic platters coated with a magnetic material, and a read/write head mounted on a mechanical arm that moves across the surface to access data. HDDs operate at rotational speeds typically ranging from 5,400 RPM for consumer-grade drives to 7,200 RPM or even 15,000 RPM for high-performance server drives. The faster the rotation speed, the quicker the drive can access data. HDDs connect to the system via the SATA interface in most modern consumer devices, or via SAS (Serial Attached SCSI) in enterprise environments, which offers higher reliability and performance. HDDs offer the highest storage capacities at the lowest cost per gigabyte, with consumer drives commonly ranging from 500 GB to 20 TB or more. This makes them a popular choice for bulk data storage, backups, and archiving. However, their mechanical nature makes them slower, noisier, more power-hungry, and vulnerable to physical shock and vibration. Average read/write speeds for a typical HDD range from 80 to 160 MB/s, with access times measured in milliseconds due to the physical movement of the read/write head.

Solid-state drive (SSD) on page 217 uses NAND flash memory chips with no moving parts, which makes it significantly faster, quieter, and more reliable than an HDD. NAND flash memory itself comes in several types that differ in performance, endurance, and cost. SLC (Single-Level Cell) stores one bit per cell and offers the highest endurance and speed, making it suitable for enterprise applications. MLC (Multi-Level Cell) stores two bits per cell, offering a balance of performance and cost. TLC (Triple-Level Cell) stores three bits per cell and is the most common type found in consumer SSDs today, offering good capacity at a lower price. QLC (Quad-Level Cell) stores four bits per cell, maximizing capacity and reducing cost further, but at the expense of write endurance and speed. SSDs are commonly connected via the SATA interface, which supports transfer speeds of up to 600 MB/s — significantly faster than an HDD but still limited by the SATA protocol itself. SSDs are widely used as the primary operating system drive in laptops and desktop computers due to their fast boot times, near-instant application loading, and improved overall system responsiveness. They are also far more resistant to physical shock since there are no moving parts. Typical consumer SATA SSDs deliver read speeds of 500–550 MB/s and write speeds of 450–520 MB/s, with access times measured in microseconds rather than milliseconds.

Non-Volatile Memory Express (NVMe) on page 212 is a modern communication protocol designed specifically for flash-based storage, built to overcome the bottlenecks of the older SATA interface. NVMe drives connect directly to the CPU via the PCIe (Peripheral Component Interconnect Express) bus, typically through an M.2 slot on the motherboard, allowing for a much higher bandwidth data path. NVMe drives using PCIe Gen 3 typically achieve read speeds of 3,000–3,500 MB/s, while PCIe Gen 4 drives push that to 5,000–7,000 MB/s, and the latest PCIe Gen 5 NVMe drives can exceed 12,000 MB/s — making them many times faster than SATA-based SSDs. In addition to raw speed, NVMe also significantly reduces latency and supports a much higher number of input/output operations per second (IOPS) compared to SATA drives, which is critical for demanding workloads such as video editing, 3D rendering, databases, and virtualization. NVMe drives are available in several form factors. The most common is M.2, a compact card that plugs directly into the motherboard. The U.2 form factor is used in enterprise environments and resembles a small 2.5-inch drive with a dedicated connector. PCIe add-in cards are also available for workstations that require maximum performance. Like SSDs, NVMe drives use NAND flash memory in SLC, MLC, TLC, or QLC configurations, with the same trade-offs between endurance, capacity, and cost. NVMe drives are the preferred choice for high-performance workstations, gaming systems, and servers where speed and low latency are critical.

USB flash drive on page 218 are small, portable storage devices that use flash memory and connect via a USB port. They are widely used for transferring files between computers, storing portable applications, and creating bootable media. Their capacities typically range from a few gigabytes to several terabytes. USB 3.0 and 3.1 drives offer transfer speeds of up to 400–600 MB/s, though many budget drives operate significantly slower. Their compact size and ease of use make them one of the most commonly used storage accessories.

Secure Digital (SD) card on page 216 are compact flash-based storage cards commonly used in digital cameras, smartphones, drones, and other portable devices. They come in various form factors — standard SD, miniSD, and microSD — and speed classes that define their minimum data transfer rates. High-speed variants such as UHS-I and UHS-II are used in professional photography and video recording, with UHS-II cards capable of read speeds up to 300 MB/s.

In many modern systems, several storage types are used together — for example, an NVMe drive for the operating system and applications, and an HDD for bulk data storage and backups — combining the speed benefits of flash-based storage with the cost efficiency of mechanical drives.

Hardware and disk organization

Understanding of underlying mechanisms of data storage, organization and data recovery.

This section provide a comprehensive overview of how hard disk drives physically store data and logically organize it for computer systems to access and boot from.

This section contains information about:

Physical Storage Layer:
Hard disks contain spinning platters with data stored in concentric tracks, divided into 512-byte sectors (the smallest storage units). Tracks at the same position across multiple platters form cylinders. File systems group sectors into clusters for efficient allocation, though this can lead to fragmentation when contiguous space is unavailable.

Boot Process Layer:
The Master Boot Record (MBR), located at the very first sector of every disk, contains executable code and the Partition Table. During startup, the MBR identifies the system partition, loads its boot sector, and transfers control to continue booting the operating system. This critical location is vulnerable to viruses that can prevent system startup.

Partition Organization Layer:
The Partition Table (within the MBR sector) defines up to four partitions per disk, specifying each partition's file system type, bootability, starting/ending locations, and size. Due to field size constraints, it can address disks up to approximately 7.8 GB directly. For more than four partitions, extended partitions create a linked chain of logical drives, each with its own partition table.

Practical Implications
Understanding these layers is essential for disk management, troubleshooting boot failures, performing data recovery, and comprehending capacity limitations in legacy systems. The architecture balances backward compatibility with MS-DOS/Windows 95 while supporting modern file systems like NTFS.

Hard Disk Drive Basics

Understanding of underlying mechanisms of data storage, organization and data recovery.

A hard disk is a sealed unit containing a number of *platters* in a stack. Hard disks may be mounted in a horizontal or a vertical position. In this description, the hard drive is mounted horizontally. Electromagnetic read/write *heads* are positioned above and below each platter. As the platters spin, the drive heads move in toward the center surface and out toward the edge. In this way, the drive heads can reach the entire surface of each platter.

Each disk consists of platters, rings on each side of each platter called tracks, and sections within each track called sectors. A sector is the smallest physical storage unit on a disk, almost always 512 bytes in size.

Figure below illustrates a hard disk with two platters. The remainder of this section describes the terms used on the figure.

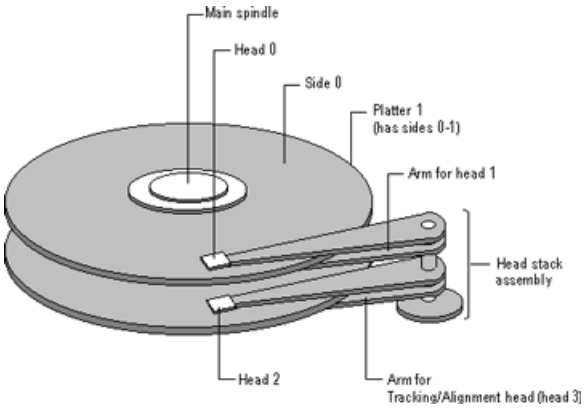


Figure 59: Two plated hard disk

The cylinder/head/sector notation scheme described in this section is slowly being eliminated. All new disks use some kind of translation factor to make their actual hardware layout appear as something else, mostly to work with MS-DOS and Windows 95.

Tracks and Cylinders

On hard disks, the data are stored on the disk in thin, concentric bands called *tracks*. There can be more than a thousand tracks on a 3½ inch hard disk. Tracks are a logical rather than physical structure, and are established when the disk is low-level formatted. Track numbers start at 0, and track 0 is the outermost track of the disk. The highest numbered track is next to the spindle. If the disk geometry is being translated, the highest numbered track would typically be 1023. Next figure shows track 0, a track in the middle of the disk, and track 1023.

A *cylinder* consists of the set of tracks that are at the same head position on the disk. In a figure below, cylinder 0 is the four tracks at the outermost edge of the sides of the platters. If the disk has 1024 cylinders (which would be numbered 0-1023), cylinder 1023 consists of all of the tracks at the innermost edge of each side.

Most disks used in personal computers today rotate at a constant angular velocity. The tracks near the outside of the disk are less densely populated with data than the tracks near the center of the disk. Thus, a fixed amount of data can be read in a constant period of time, even though the speed of the disk surface is faster on the tracks located further away from the center of the disk.

Modern disks reserve one side of one platter for track positioning information, which is written to the disk at the factory during disk assembly. It is not available to the operating system. The disk controller uses this information to fine tune the head locations when the heads move to another location on the disk. When a side contains the track position information, that side cannot be used for data. Thus, a disk assembly containing two platters has three sides that are available for data.

Sectors and Clusters

Each track is divided into sections called *sectors*. A sector is the smallest physical storage unit on the disk. The data size of a sector is always a power of two, and is almost always 512 bytes.

Each track has the same number of sectors, which means that the sectors are packed much closer together on tracks near the center of the disk. Next figure shows sectors on a track. You can see that sectors closer to the spindle are closer together than those on the outside edge of the disk. The disk controller uses the sector identification information stored in the area immediately before the data in the sector to determine where the sector itself begins.

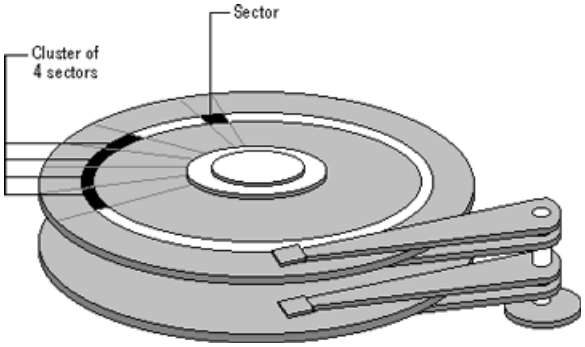


Figure 60: Clusters and sectors

As a file is written to the disk, the file system allocates the appropriate number of *clusters* to store the file's data. For example, if each cluster is 512 bytes and the file is 800 bytes, two clusters are allocated for the file. Later, if you update the file to, for example, twice its size (1600 bytes), another two clusters are allocated.

If contiguous clusters (clusters that are next to each other on the disk) are not available, the data are written elsewhere on the disk, and the file is considered to be *fragmented*. Fragmentation is a problem when the file system must search several different locations to find all the pieces of the file you want to read. The search causes a delay before the file is retrieved. A larger cluster size reduces the potential for fragmentation, but increases the likelihood that clusters will have unused space.

Using clusters larger than one sector reduces fragmentation, and reduces the amount of disk space needed to store the information about the used and unused areas on the disk.

The stack of platters rotate at a constant speed. The drive head, while positioned close to the center of the disk reads from a surface that is passing by more slowly than the surface at the outer edges of the disk. To compensate for this physical difference, tracks near the outside of the disk are less-densely populated with data than the tracks near the center of the disk. The result of the different data density is that the same amount of data can be read over the same period of time, from any drive head position.

The disk space is filled with data according to a standard plan. One side of one platter contains space reserved for hardware track-positioning information and is not available to the operating system. Thus, a disk assembly containing two platters has three sides available for data. Track-positioning data is written to the disk during assembly at the factory. The system *disk controller* reads this data to place the drive heads in the correct sector position.

Related concepts

Hardware and disk organization on page 129
Understanding of underlying mechanisms of data storage, organization and data recovery.

Master Boot Record (MBR) on page 132
Understanding of underlying mechanisms of data storage, organization and data recovery.

Partition table on page 133

Understanding of underlying mechanisms of data storage, organization and data recovery.

Master Boot Record (MBR)

Understanding of underlying mechanisms of data storage, organization and data recovery.

The Master Boot Record, created when you create the first partition on the hard disk, is probably the most important data structure on the disk. It is the first sector on every disk. The location is always track (cylinder) 0, side (head) 0, and sector 1.

The Master Boot Record contains the Partition table on page 133 for the disk and a small amount of executable code. On x86-based computers, the executable code examines the Partition Table, and identifies the system partition. The Master Boot Record then finds the system partition's starting location on the disk, and loads an copy of its Partition Boot Sector into memory. The Master Boot Record then transfers execution to executable code in the Partition Boot Sector.

Note:

Although there is a Master Boot Record on every hard disk, the executable code in the sector is used only if the disk is connected to an x86-based computer and the disk contains the system partition.

Figure below shows a hex dump of the sector containing the Master Boot Record. The figure shows the sector in two parts. The first part is the Master Boot Record, which occupies the first 446 bytes of the sector. The disk signature (FD 4E F2 14) is at the end of the Master Boot Record code. The second part is the Partition table on page 133.

```
Physical Sector: Cyl 0, Side 0, Sector 1

00000000: 00 33 C0 8E D0 BC 00 7C  -  8B F4 50 07 50 1F FB FC   .3.....|..P.P..
00000010: BF 00 06 B9 00 01 F2 A5  -  EA 1D 06 00 00 BE BE 07   ................
00000020: B3 04 80 3C 80 74 0E 80  -  3C 00 75 1C 83 C6 10 FE   ...<.t..<.u.....
00000030: CB 75 EF CD 18 8B 14 8B  -  4C 02 8B EE 83 C6 10 FE   .u......L.......
00000040: CB 74 1A 80 3C 00 74 F4  -  BE 8B 06 AC 3C 00 74 0B   .t..<.t.....<.t.
00000050: 56 BB 07 00 B4 0E CD 10  -  5E EB F0 EB FE BF 05 00   V.......^.......
00000060: BB 00 7C B8 01 02 57 CD  -  13 5F 73 0C 33 C0 CD 13   ..|...W.._s.3...
00000070: 4F 75 ED BE A3 06 EB D3  -  BE C2 06 BF FE 7D 81 3D   Ou...........}.=
00000080: 55 AA 75 C7 8B F5 EA 00  -  7C 00 00 49 6E 76 61 6C   U.u.....|..Inval
00000090: 69 64 20 70 61 72 74 69  -  74 69 6F 6E 20 74 61 62   id partition tab
000000A0: 6C 65 00 45 72 72 6F 72  -  20 6C 6F 61 64 69 6E 67   le.Error loading
000000B0: 20 6F 70 65 72 61 74 69  -  6E 67 20 73 79 73 74 65    operating syste
000000C0: 6D 00 4D 69 73 73 69 6E  -  67 20 6F 70 65 72 61 74   m.Missing operat
000000D0: 69 6E 67 20 73 79 73 74  -  65 6D 00 00 80 45 14 15   ing system...E..
000000E0: 00 00 00 00 00 00 00 00  -  00 00 00 00 00 00 00 00   ................
000000F0: 00 00 00 00 00 00 00 00  -  00 00 00 00 00 00 00 00   ................
00000100: 00 00 00 00 00 00 00 00  -  00 00 00 00 00 00 00 00   ................
00000110: 00 00 00 00 00 00 00 00  -  00 00 00 00 00 00 00 00   ................
00000120: 00 00 00 00 00 00 00 00  -  00 00 00 00 00 00 00 00   ................
00000130: 00 00 00 00 00 00 00 00  -  00 00 00 00 00 00 00 00   ................
00000140: 00 00 00 00 00 00 00 00  -  00 00 00 00 00 00 00 00   ................
00000150: 00 00 00 00 00 00 00 00  -  00 00 00 00 00 00 00 00   ................
00000160: 00 00 00 00 00 00 00 00  -  00 00 00 00 00 00 00 00   ................
00000170: 00 00 00 00 00 00 00 00  -  00 00 00 00 00 00 00 00   ................
00000180: 00 00 00 00 00 00 00 00  -  00 00 00 00 00 00 00 00   ................
00000190: 00 00 00 00 00 00 00 00  -  00 00 00 00 00 00 00 00   ................
000001A0: 00 00 00 00 00 00 00 00  -  00 00 00 00 00 00 00 00   ................
000001B0: 00 00 00 00 00 00 00 00  -  FD 4E F2 14 00 00 80 01   .........N......
000001C0: 01 00 06 0F 7F 96 3F 00  -  00 00 51 42 06 00 00 00   ....#.?...QB....
000001D0: 41 97 07 0F FF 2C 90 42  -  06 00 A0 3E 06 00 00 00   A....,.B...>....
000001E0: C1 2D 05 0F FF 92 30 81  -  0C 00 A0 91 01 00 00 00   .-....0.........
000001F0: C1 93 01 0F FF A6 D0 12  -  0E 00 C0 4E 00 00 55 AA   ...........N..U.
```

Important:

Viruses Can Infect the Master Boot Record!

Many destructive viruses damage the Master Boot Record and make it impossible to start the computer from the hard disk. Because the code in the Master Boot Record executes before any operating system is started, no operating system can detect or recover from corruption of the Master Boot Record. You can use, for example, the DiskProbe program on *Windows NT Workstation Resource Kit* CD to display the Master Boot Record, and compare it to the Master Boot Record

shown above. There are also utilities on the Microsoft Windows Resource Kits that enable you to save and restore the Master Boot Record.

Tip:

For more detailed information see resource kits on Microsoft's web site http://www.microsoft.com/windows/reskits/webresources/default.asp or Microsoft Developers Network (MSDN) http://msdn.microsoft.com

Related concepts

Hardware and disk organization on page 129
Understanding of underlying mechanisms of data storage, organization and data recovery.

Hard Disk Drive Basics on page 129
Understanding of underlying mechanisms of data storage, organization and data recovery.

Partition table on page 133
Understanding of underlying mechanisms of data storage, organization and data recovery.

Partition table

Understanding of underlying mechanisms of data storage, organization and data recovery.

The information about primary partitions and an extended partition is contained in the Partition Table, a 64-byte data structure located in the same sector as the Master Boot Record (MBR) on page 132 (cylinder 0, head 0, sector 1). The Partition Table conforms to a standard layout that is independent of the operating system. Each Partition Table entry is 16 bytes long, making a maximum of four entries available. Each entry starts at a predetermined offset from the beginning of the sector, as follows:

- Partition 1 0x01BE (446)
- Partition 2 0x01CE (462)
- Partition 3 0x01DE (478)
- Partition 4 0x01EE (494)

The last two bytes in the sector are a signature word for the sector and are always 0x55AA.

The next figure is a printout of the Partition Table for the disk shown in a Master Boot Record (MBR) on page 132 earlier in this chapter. When there are fewer than four partitions, the remaining fields are all zeros.

```
000001B0:                                                 80 01                ..
000001C0: 01 00 06 0F 7F 96 3F 00 - 00 00 51 42 06 00 00 00   ....#.?...QB.....
000001D0: 41 97 07 0F FF 2C 90 42 - 06 00 A0 3E 06 00 00 00   A....,.B...>....
000001E0: C1 2D 05 0F FF 92 30 81 - 0C 00 A0 91 01 00 00 00   .-....0.........
000001F0: C1 93 01 0F FF A6 D0 12 - 0E 00 C0 4E 00 00 55 AA   ...........N..U.
```

The following table describes each entry in the Partition Table. The sample values correspond to the information for partition 1.

Table 2: Partition Table Fields

Byte Offset	Field Length	Sample Value	Meaning
00	BYTE	0x80	Boot Indicator. Indicates whether the partition is the system partition. Legal values are: 00 = Do not use for booting. 80 = System partition.
01	BYTE	0x01	Starting Head.

Byte Offset	Field Length	Sample Value	Meaning
02	6 bits	0x01	Starting Sector. Only bits 0-5 are used. Bits 6-7 are the upper two bits for the Starting Cylinder field.
03	10 bits	0x00	Starting Cylinder. This field contains the lower 8 bits of the cylinder value. Starting cylinder is thus a 10-bit number, with a maximum value of 1023.
04	BYTE	0x06	System ID. This byte defines the volume type. In Windows NT, it also indicates that a partition is part of a volume that requires the use of the HKEY_LOCAL_MACHINE \SYSTEM\DISK Registry subkey.
05	BYTE	0x0F	Ending Head.
06	6 bits	0x3F	Ending Sector. Only bits 0-5 are used. Bits 6-7 are the upper two bits for the Ending Cylinder field.
07	10 bits	0x196	Ending Cylinder. This field contains the lower 8 bits of the cylinder value. Ending cylinder is thus a 10-bit number, with a maximum value of 1023.
08	DWORD	3F 00 00 00	Relative Sector.
12	DWORD	51 42 06 00	Total Sectors.

The remainder of this section describes the uses of these fields. Definitions of the fields in the Partition Table is the same for primary partitions, extended partitions, and logical drives in extended partitions.

Boot Indicator Field

The Boot Indicator field indicates whether the volume is the system partition. On x-86-based computers, only one primary partition on the disk should have this field set. This field is used only on x86-based computers. On RISC-based computers, the NVRAM contains the information for finding the files to load.

On x86-based computers, it is possible to have different operating systems and different file systems on different volumes. For example, a computer could have MS-DOS on the first primary partition and Windows 95, UNIX, OS/2, or Windows NT on the second. You control which primary partition (active partition in FDISK) to use to start the computer by setting the Boot Indicator field for that partition in the Partition Table.

System ID field

For primary partitions and logical drives, the System ID field describes the file system used to format the volume. Windows NT uses this field to determine what file system device drivers to load during startup. It also identifies the extended partition, if there is one defined.

Table 3: System ID field description

Value	Meaning
0x01	12-bit FAT primary partition or logical drive. The number of sectors in the volume is fewer than 32680.
0x04	16-bit FAT primary partition or logical drive. The number of sectors is between 32680 and 65535.
0x05	Extended partition. See section titled "Logical Drives and Extended Partitions," presented later in this chapter, for more information.
0x06	BIGDOS FAT primary partition or logical drive.

Value	Meaning
0x07	NTFS primary partition or logical drive.

Figure presented earlier in this section, has examples of a BIGDOS FAT partition, an NTFS partition, an extended partition, and a 12-bit FAT partition.

If you install Windows NT on a computer that has Windows 95 preinstalled, the FAT partitions might be shown as unknown. If you want to be able to use these partitions when running Windows NT, your only option is to delete the partitions.

OEM versions of Windows 95 support the following four partition types for FAT file systems that Windows NT cannot recognize.

Value	Meaning
0x0B	Primary Fat32 partition, using interrupt 13 (INT 13) extensions.
0x0C	Extended Fat32 partition, using INT 13 extensions.
0x0E	Extended Fat16 partition, using INT 13 extensions.
0x0F	Primary Fat16 partition, using INT 13 extensions.

When you create a volume set or a stripe set, Disk Administrator sets the high bit of the System ID field for each primary partition or logical drive that is a member of the volume. For example, a FAT primary partition or logical drive that is a member of a volume set or a stripe set has a System ID value of 0x86. An NTFS primary partition or logical drive has a System ID value of 0x87. This bit indicates that Windows NT needs to use the HKEY_LOCAL_MACHINE\SYSTEM\DISK Registry subkey to determine how the members of the volume set or stripe set relate to each other. Volumes that have the high bit set can only be accessed by Windows NT.

When a primary partition or logical drive that is a member of a volume set or a stripe set has failed due to write errors or cannot be accessed, the second most significant bit is set. The System ID byte is set to C6 in the case of a FAT volume, or C7 in the case of an NTFS volume.

Note:

If you start up MS-DOS, it can only access primary partitions or logical drives that have a value of 0x01, 0x04, 0x05, or 0x06 for the System ID. However, you should be able to delete volumes that have the other values. If you use a MS-DOS-based low-level disk editor, you can read and write any sector, including ones that are in NTFS volumes.

On Windows NT Server, mirror sets and stripe sets with parity also require the use of the Registry subkey HKEY_LOCAL_MACHINE\SYSTEM\DISK to determine how to access the disks.

Starting and Ending Head, Sector, and Cylinder Fields

On x86-based computers, the Starting and Ending Head, Cylinder, and Sector fields on the start-up disk are very important for starting up the computer. The code in the Master Boot Record uses these fields to find and load the Partition Boot Sector.

The Ending Cylinder field in the Partition Table is ten bits long, which limits the maximum number of cylinders that can be described in the Partition Table to 1024. The Starting and Ending Head fields are one byte long, which limits this field to the range 0 – 255. The Starting and Ending Sector field is 6 bits long, limiting its range to 0 – 63. However, sectors start counting at 1 (versus 0 for the other fields), so the maximum number of sectors per track is 63.

Since current hard disks are low-level formatted with the industry standard 512-byte sector size, the maximum capacity disk that can be described by the Partition Table can be calculated as follows:

```
MaxCapacity = (sector size) x (sectors per track) x (cylinders) x (heads)
```

Substituting the maximum possible values yields:

```
512 x 63 x 1024 x 256 = 8,455,716,864 bytes or 7.8 GB
```

The maximum formatted capacity is slightly less than 8 GB.

However, the maximum cluster size that you can use for FAT volumes when running Windows NT is 64K, when using a 512 byte sector size. Therefore, the maximum size for a FAT volume is 4 GB.

If you have a dual-boot configuration with Windows 95 or MS-DOS, FAT volumes that might be accessed when using either of those operating systems are limited to 2 GB. In addition, Macintosh computers that are viewing volumes on a computer running Windows NT cannot see more than 2 GB. If you try to use a FAT volume larger than 2 GB when running MS-DOS or Windows 95, or access it from a Macintosh computer, you might get a message that there are 0 bytes available. The same limit applies to OS/2 system and boot partitions.

The maximum size of a FAT volume on a specific computer depends on the disk geometry, and the maximum values that can fit in the fields described in this section. The next table shows the typical size of a FAT volume when translation is enabled, and when it is disabled. The number of cylinders in both situations is 1024.

Translation mode	Numb of heads	Sector per track	Maximum size for system or boot partition
Disabled	64	32	1 GB
Enabled	255	63	4 GB

Note:

RISC-based computers do not have a limit on the size of the system or boot partitions.

If a primary partition or logical drive extends beyond cylinder 1023, all of these fields will contain the maximum values.

Relative Sectors and Number of Sectors Fields

For primary partitions, the Relative Sectors field represents the offset from the beginning of the disk to the beginning of the partition, counting by sectors. The Number of Sectors field represents the total number of sectors in the partition. For a description of these fields in extended partitions, see the section Logical Drives and Extended Partitions.

Windows NT uses these fields to access all partitions. When you format a partition when running Windows NT, it puts data into the Starting and Ending Cylinder, Head, and Sector fields only for backward compatibility with MS-DOS and Windows 95, and to maintain compatibility with the BIOS interrupt (INT) 13 for start-up purposes.

Logical Drives and Extended Partitions

When more than four logical disks are required on a single physical disk, the first partition should be a primary partition. The second partition can be created as an extended partition, which can contain all the remaining unpartitioned space on the disk.

Note:

A primary partition is one that can be used as the system partition. If the disk does not contain a system partition, you can configure the entire disk as a single, extended partition.

Some computers create an EISA configuration partition as the first partition on the hard disk.

Windows NT detects an extended partition because the System ID byte in the Partition Table entry is set to 5. There can be only one extended partition on a hard disk.

Within the extended partition, you can create any number of logical drives. As a practical matter, the number of available drive letters is the limiting factor in the number of logical drives that you can define.

When you have an extended partition on the hard disk, the entry for that partition in the Partition Table (at the end of the Master Boot Record) points to the first disk sector in the extended partition. The first sector of each logical drive in an extended partition also has a Partition Table, which is the last 66 bytes of the sector. (The last two bytes of the sector are the end-of-sector marker.)

These are the entries in an extended Partition Table:

- The first entry is for the current logical drive.
- The second entry contains information about the next logical drive in the extended partition.
- Entries three and four are all zeroes.

This format repeats for every logical drive. The last logical drive has only its own partition entry listed. The entries for partitions 2-4 are all zeroes.

The Partition Table entry is the only information on the first side of the first cylinder of each logical drive in the extended partition. The entry for partition 1 in each Partition Table contains the starting address for data on the current logical drive. And the entry for partition 2 is the address of the sector that contains the Partition Table for the next logical drive.

The use of the Relative Sector and Total Sectors fields for logical drives in an extended partition is different than for primary partitions. For the partition 1 entry of each logical drive, the Relative Sectors field is the sector from the beginning of the logical drive that contains the Partition Boot Sector. The Total Sectors field is the number of sectors from the Partition Boot Sector to the end of the logical drive.

For the partition 2 entry, the Relative Sectors field is the offset from the beginning of the extended partition to the sector containing the Partition Table for the logical drive defined in the Partition 2 entry. The Total Sectors field is the total size of the logical drive defined in the Partition 2 entry.

Note:

If a logical drive is part of a volume set, the Partition Boot Sector is at the beginning of the first member of the volume set. Other members of the volume set have data where the Partition Boot Sector would normally be located.

Tip:

For more detailed information see resource kits on Microsoft's web site http://www.microsoft.com/windows/reskits/webresources/default.asp or Microsoft Developers Network (MSDN) http://msdn.microsoft.com

Related concepts

Hardware and disk organization on page 129
Understanding of underlying mechanisms of data storage, organization and data recovery.

Hard Disk Drive Basics on page 129
Understanding of underlying mechanisms of data storage, organization and data recovery.

Master Boot Record (MBR) on page 132

Understanding of underlying mechanisms of data storage, organization and data recovery.

Disk arrays (RAID)

Redundant array of independent disks (RAID)

Redundant array of independent disks (RAID) is a storage technology that combines multiple disk drive components into a logical unit. Data is distributed across the drives in one of several ways called "RAID levels", depending on what level of redundancy and performance (via parallel communication) is required.

RAID-0
This technique has striping but no redundancy of data. It offers the best performance but no fault-tolerance.

RAID-1
This type is also known as disk mirroring and consists of at least two drives that duplicate the storage of data. There is no striping. Read performance is improved since either disk can be read at the same time. Write performance is the same as for single disk storage. RAID-1 provides the best performance and the best fault-tolerance in a multi-user system.

RAID-2
This type uses striping across disks with some disks storing error checking and correcting (ECC) information. It has no advantage over RAID-3.

RAID-3
This type uses striping and dedicates one drive to storing parity information. The embedded error checking (ECC) information is used to detect errors. Data recovery is accomplished by calculating the exclusive OR (XOR) of the information recorded on the other drives. Since an I/O operation addresses all drives at the same time, RAID-3 cannot overlap I/O. For this reason, RAID-3 is best for single-user systems with long record applications.

RAID-4
This type uses large stripes, which means you can read records from any single drive. This allows you to take advantage of overlapped I/O for read operations. Since all write operations have to update the parity drive, no I/O overlapping is possible. RAID-4 offers no advantage over RAID-5.

RAID-5
This type includes a rotating parity array, thus addressing the write limitation in RAID-4. Thus, all read and write operations can be overlapped. RAID-5 stores parity information but not redundant data (but parity information can be used to reconstruct data). RAID-5 requires at least three and usually five disks for the array. It's best for multi-user systems in which performance is not critical or which do few write operations.

Parity tables

Left Synchronous			
0	5	6	P
1	4	P	11
2	P	7	10
P	3	8	9

Left Asynchronous			
0	3	6	P
1	4	P	9

Left Asynchronous			
2	P	7	10
P	5	8	11

Right Synchronous			
P	5	6	11
0	P	7	10
1	4	P	9
2	3	8	P

Right Asynchronous			
P	3	6	9
0	P	7	10
1	4	P	11
2	5	8	P

Logical Disk Manager

Understanding of underlying mechanisms of data storage, organization and data recovery.

Dynamic disks provide features that basic disks do not, such as the ability to create volumes that span multiple disks (spanned and striped volumes), and the ability to create fault tolerant volumes (mirrored and RAID-5 volumes). All volumes on dynamic disks are known as dynamic volumes.

There are five types of dynamic volumes:

Simple
A dynamic volume made up of disk space from a single dynamic disk. A simple volume can consist of a single region on a disk or multiple regions of the same disk that are linked together. If the simple volume is not a system volume or boot volume, you can extend it within the same disk or onto additional disks. If you extend a simple volume across multiple disks, it becomes a spanned volume. You can create simple volumes only on dynamic disks. Simple volumes are not fault tolerant, but you can mirror them to create mirrored volumes on computers running the Windows 2000 Server or Windows Server 2003 families of operating systems.

Spanned
A dynamic volume consisting of disk space on more than one physical disk. You can increase the size of a spanned volume by extending it onto additional dynamic disks. You can create spanned volumes only on dynamic disks. Spanned volumes are not fault tolerant and cannot be mirrored.

Striped
A dynamic volume that stores data in stripes on two or more physical disks. Data in a striped volume is allocated alternately and evenly (in stripes) across the disks. Striped volumes offer the best performance of all the volumes that are available in Windows, but they do not provide fault tolerance. If a disk in a striped volume fails, the data in the entire volume is lost. You can create striped volumes only on dynamic disks. Striped volumes cannot be mirrored or extended.

Mirrored
A fault-tolerant volume that duplicates data on two physical disks. A mirrored volume provides data redundancy by using two identical volumes, which are called mirrors, to duplicate the information contained on the volume. A mirror is always located on a different disk. If one of the physical

disks fails, the data on the failed disk becomes unavailable, but the system continues to operate in the mirror on the remaining disk. You can create mirrored volumes only on dynamic disks on computers running the Windows 2000 Server or Windows Server 2003 families of operating systems. You cannot extend mirrored volumes.

RAID-5
A fault-tolerant volume with data and parity striped intermittently across three or more physical disks. Parity is a calculated value that is used to reconstruct data after a failure. If a portion of a physical disk fails, Windows recreates the data that was on the failed portion from the remaining data and parity. You can create RAID-5 volumes only on dynamic disks on computers running the Windows 2000 Server or Windows Server 2003 families of operating systems. You cannot mirror or extend RAID-5 volumes. In Windows NT 4.0, a RAID-5 volume was known as a striped set with parity.

Mirrored and RAID-5 volumes are fault tolerant and are available only on computers running Windows 2000 Server, Windows 2000 Advanced Server, Windows 2000 Datacenter Server, or the Windows Server 2003 family of operating systems. You can, however, use a computer running Windows XP Professional to remotely create mirrored and RAID-5 volumes on these operating systems.

Regardless of whether the dynamic disk uses the master boot record (MBR) or GUID partition table (GPT) partition style, you can create up to 2,000 dynamic volumes, although the recommended number of dynamic volumes is 32 or less.

For information about how to manage dynamic volumes, see Manage dynamic volumes.

Virtual Disks

Active@ KillDisk provides full support for Virtual Disks - dynamic disks created and managed by:

- ***Logical Disk Manager*** (LDM on Windows)
- ***Logical Volume Manager*** (LVM on Linux)
- ***Windows Storage Spaces*** (WSS on Windows)

Virtual Disks are virtual devices which look like regular physical disks to all applications. These virtual devices are stored on one or more physical disks and emulate different types of volumes and RAID disk arrays not on a hardware level (inside disk controller), but on Operating System level (software emulation). Virtual devices are fully supported by the **Active@ KillDisk**. These disks will appear in ***Local Devices*** view like any other regular disks. When you launch an erase for the virtual disk, the progress is displayed in the same color on all components of the composite virtual drive.

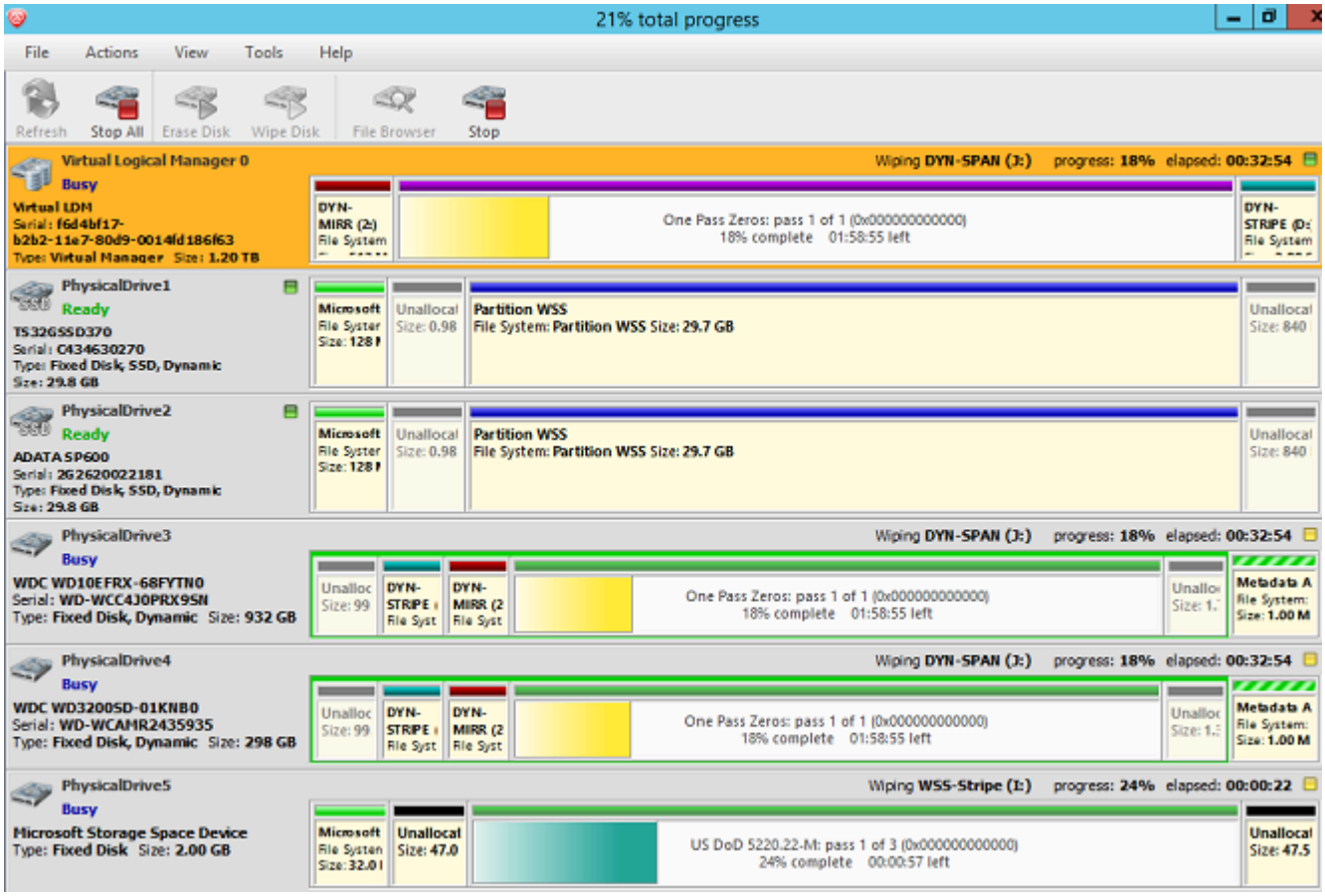


Figure 61: Erasing a Virtual Drive (Striped Disk Array)

Note:

By default Virtual Disks are not being displayed in the list of devices. To display Virtual Disks go to ***Preferences*** > ***General Settings*** and turn on ***Initialize virtual disks*** option.

File Systems

File Systems overview

A file system is a structured method used by operating systems to organize, store, manage, and retrieve data on storage devices such as hard drives, solid-state drives, USB flash drives, and removable media. It serves as the foundation for how data is written to and read from storage media, acting as an intermediary layer between the physical hardware and the operating system.

At its core, a file system defines the logical structure of stored data by establishing rules for file naming conventions, directory hierarchies, metadata management, and space allocation. It maintains critical information about each file, including its name, size, location on the disk, creation and modification dates, and access permissions. The file system tracks which sectors or blocks of the storage device are occupied and which are available for new data, ensuring efficient use of storage space and preventing data corruption.

File systems also implement various mechanisms for data integrity and reliability. These include journaling capabilities that record changes before they are committed to the disk, error detection and correction algorithms, and support for features such as file compression, encryption, and access control. Different file systems are optimized for specific use cases, balancing factors such as performance, compatibility, reliability, and support for advanced features.

Windows NT File System (NTFS) on page 155
The primary file system for modern Windows operating systems. NTFS supports large file sizes, advanced security features, file compression, and encryption. It is widely used for internal hard drives, external drives, and enterprise storage solutions on Windows platforms.

FAT File System on page 142
A legacy file system compatible with most operating systems, including Windows, macOS, and Linux. FAT32 is commonly used for USB flash drives, memory cards, and external storage devices that require cross-platform compatibility. However, it has limitations, including a maximum file size of 4 GB.

Extended File System (exFAT) on page 167
Developed as an improvement over FAT32, exFAT supports larger file sizes and is optimized for flash drives and SD cards. It maintains broad compatibility across Windows, macOS, and many other devices, making it ideal for portable storage media.

Fourth Extended File System (Ext4 on page 208)
The default file system for most Linux distributions. ext4 provides high performance, reliability, and support for large volumes and files. It is primarily used on Linux-based systems for both system drives and data storage.

Apple File System (ApFS) on page 167
The modern file system used by macOS, iOS, and other Apple devices. APFS is optimized for solid-state drives and flash storage, offering features such as encryption, snapshots, and space sharing. It replaced HFS+ as the default file system for Apple platforms.

Hierarchical File System (HFS and HFS+)
The predecessor to APFS, HFS+ was used by macOS for many years. While largely replaced by APFS on newer systems, it is still found on older Mac computers and external drives formatted for macOS compatibility.

- Hierarchical File System (HFS) on page 165
- Hierarchical File System Plus (HFS+) on page 166

FAT File System

Understanding of underlying mechanisms of data storage, organization and data recovery.

The FAT file system is a simple file system originally designed for small disks and simple folder structures. The FAT file system is named for its method of organization, the File Allocation Table, which resides at the beginning of the volume. To protect the volume, two copies of the table are kept, in case one becomes damaged. In addition, the file allocation tables and the root folder must be stored in a fixed location so that the files needed to start the system can be correctly located.

A volume formatted with the FAT file system is allocated in clusters. The default cluster size is determined by the size of the volume. For the FAT file system, the cluster number must fit in 16 bits and must be a power of two.

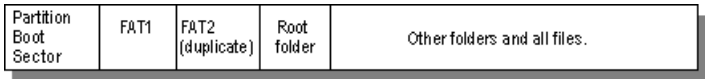


Figure 62: FAT file system volume organization

See the next sections for more information about FAT:

- FAT Partition Boot Sector on page 143
- File Allocation Table (FAT) on page 145
- FAT Root Folder on page 146
- FAT Folder Structure on page 146
- FAT32 Features on page 147

Main differences between FAT12, FAT16, FAT32

- FAT12 file system contains 1.5 bytes per cluster within the file allocation table.
- FAT16 file system contains 2 bytes per cluster within the file allocation table.
- FAT32 file system includes 4 bytes per cluster within the file allocation table.

Related concepts

Windows NT File System (NTFS) on page 155
Understanding of underlying mechanisms of data storage, organization and data recovery.

Extended File System (exFAT) on page 167
Understanding of underlying mechanisms of data storage, organization and data recovery.

FAT Partition Boot Sector

Understanding of underlying mechanisms of data storage, organization and data recovery.

The Partition Boot Sector contains information that the file system uses to access the volume. On x86-based computers, the Master Boot Record use the Partition Boot Sector on the system partition to load the operating system kernel files.

Next table describes the fields in the Partition Boot Sector for a volume formatted with the FAT file system.

Table 4: System ID field description

Byte Offset (in hex)	Field Length	Sample Value	Meaning
00	3 bytes	EB 3C 90	Jump instruction
03	8 bytes	MSDOS5.0	OEM Name in text
0B	25 bytes		BIOS Parameter Block
24	26 bytes		Extended BIOS Parameter Block

Byte Offset (in hex)	Field Length	Sample Value	Meaning
3E	448 bytes		Bootstrap code
1FE	2 bytes	0x55AA	End of sector marker

Table 5: BIOS Parameter Block and Extended BIOS Parameter Block Fields

Byte Offset	Field Length	Sample Value	Meaning
0x0B	WORD	0x0002	Bytes per Sector. The size of a hardware sector. For most disks in use in the United States, the value of this field is 512.
0x0D	BYTE	0x08	Sectors Per Cluster. The number of sectors in a cluster. The default cluster size for a volume depends on the volume size and the file system.
0x0E	WORD	0x0100	Reserved Sectors. The number of sectors from the Partition Boot Sector to the start of the first file allocation table, including the Partition Boot Sector. The minimum value is 1. If the value is greater than 1, it means that the bootstrap code is too long to fit completely in the Partition Boot Sector.
0x10	BYTE	0x02	Number of file allocation tables (FATs). The number of copies of the file allocation table on the volume. Typically, the value of this field is 2.
0x11	WORD	0x0002	Root Entries. The total number of file name entries that can be stored in the root folder of the volume. One entry is always used as a Volume Label. Files with long file names use up multiple entries per file. Therefore, the largest number of files in the root folder is typically 511, but you will run out of entries sooner if you use long file names.
0x13	WORD	0x0000	Small Sectors. The number of sectors on the volume if the number fits in 16 bits (65535). For volumes larger than 65536 sectors, this field has a value of 0 and the Large Sectors field is used instead.
0x15	BYTE	0xF8	Media Type. Provides information about the media being used. A value of 0xF8 indicates a hard disk.
0x16	WORD	0xC900	Sectors per file allocation table (FAT). Number of sectors occupied by each of the file allocation tables on the volume. By using this information, together with the Number of FATs and Reserved Sectors, you can compute where the root folder begins. By using the number of entries in the root folder, you can also compute where the user data area of the volume begins.
0x18	WORD	0x3F00	Sectors per Track. The apparent disk geometry in use when the disk was low-level formatted.
0x1A	WORD	0x1000	Number of Heads. The apparent disk geometry in use when the disk was low-level formatted.
0x1C	DWORD	3F 00 00 00	Hidden Sectors. Same as the Relative Sector field in the Partition Table.
0x20	DWORD	51 42 06 00	Large Sectors. If the Small Sectors field is zero, this field contains the total number of sectors in the volume. If Small Sectors is nonzero, this field contains zero..

Byte Offset	Field Length	Sample Value	Meaning
0x24	BYTE	0x80	Physical Disk Number. This is related to the BIOS physical disk number. Floppy drives are numbered starting with 0x00 for the A disk. Physical hard disks are numbered starting with 0x80. The value is typically 0x80 for hard disks, regardless of how many physical disk drives exist, because the value is only relevant if the device is the startup disk.
0x25	BYTE	0x00	Current Head. Not used by the FAT file system.
0x26	BYTE	0x29	Signature. Must be either 0x28 or 0x29 in order to be recognized by Windows NT.
0x27	4 bytes	CE 13 46 30	Volume Serial Number. A unique number that is created when you format the volume.
0x2B	11 bytes	NO NAME	Volume Label. This field was used to store the volume label, but the volume label is now stored as special file in the root directory.
0x36	8 bytes	FAT16	System ID. Either FAT12 or FAT16, depending on the format of the disk.

Tip:

For more detailed information see resource kits on Microsoft's web site http://www.microsoft.com/windows/reskits/webresources/default.asp or Microsoft Developers Network (MSDN) http://msdn.microsoft.com

File Allocation Table (FAT)

Understanding of underlying mechanisms of data storage, organization and data recovery.

The FAT file system is named for its method of organization, the file allocation table, which resides at the beginning of the volume. To protect the volume, two copies of the table are kept, in case one becomes damaged. In addition, the file allocation tables must be stored in a fixed location so that the files needed to start the system can be correctly located.

The file allocation table contains the following types of information about each cluster on the volume (see example below for FAT16):

- Unused (0x0000)
- Cluster in use by a file
- Bad cluster (0xFFF7)
- Last cluster in a file (0xFFF8-0xFFFF)

There is no organization to the FAT folder structure, and files are given the first available location on the volume. The starting cluster number is the address of the first cluster used by the file. Each cluster contains a pointer to the next cluster in the file, or an indication (0xFFFF) that this cluster is the end of the file. These links and end of file indicators are shown below.

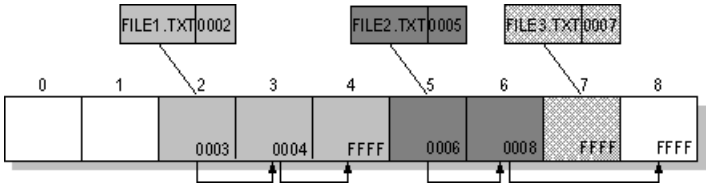


Figure 63: Example of File Allocation Table

This illustration shows three files. The file File1.txt is a file that is large enough to use three clusters. The second file, File2.txt, is a fragmented file that also requires three clusters. A small file, File3.txt, fits completely in one cluster. In each case, the folder entry (see folder entry for details) points to the first cluster of the file.

Tip:

For more detailed information see resource kits on Microsoft's web site http://www.microsoft.com/windows/reskits/webresources/default.asp or Microsoft Developers Network (MSDN) http://msdn.microsoft.com

FAT Root Folder

Understanding of underlying mechanisms of data storage, organization and data recovery.

The root folder contains an entry for each file and folder on the root. The only difference between the root folder and other folders is that the root folder is on a specified location on the disk and has a fixed size (512 entries for a hard disk, number of entries on a floppy disk depends on the size of the disk).

See FAT Folder Structure on page 146 topic for details about folder organization.

Tip:

For more detailed information see resource kits on Microsoft's web site http://www.microsoft.com/windows/reskits/webresources/default.asp or Microsoft Developers Network (MSDN) http://msdn.microsoft.com

FAT Folder Structure

Understanding of underlying mechanisms of data storage, organization and data recovery.

Folders have set of 32-byte *Folder Entries* for each file and sub-folder contained in the folder (see example figure below).

The *Folder Entry* includes the following information:

- Name (eight-plus-three characters)
- Attribute byte (8 bits worth of information, described later in this section)
- Create time (24 bits)
- Create date (16 bits)
- Last access date (16 bits)
- Last modified time (16 bits)
- Last modified date (16 bits.)
- Starting cluster number in the file allocation table (16 bits)
- File size (32 bits)

There is no organization to the FAT folder structure, and files are given the first available location on the volume. The starting cluster number is the address of the first cluster used by the file. Each cluster contains a pointer to the next cluster in the file, or an indication (0xFFFF) that this cluster is the end of the file. See File Allocation Table (FAT) on page 145 for details.

The information in the folder is used by all operating systems that support the FAT file system. In addition, Windows NT can store additional time stamps in a FAT folder entry. These time stamps show when the file was created or last accessed and are used principally by POSIX applications.

Because all entries in a folder are the same size, the attribute byte for each entry in a folder describes what kind of entry it is. One bit indicates that the entry is for a sub folder, while another bit marks the entry as a volume label. Normally, only the operating system controls the settings of these bits.

A FAT file has four attributes bits that can be turned on or off by the user — archive file, system file, hidden file, and read-only file.

File names on FAT Volumes

Beginning with Windows NT 3.5, files created or renamed on FAT volumes use the attribute bits to support long file names in a way that does not interfere with how MS-DOS or OS/2 accesses the volume. Whenever a user creates a file with a long file name, Windows creates an eight-plus-three name for the file. In addition to this conventional entry, Windows creates one or more secondary folder entries for the file, one for each 13 characters in the long file name. Each of these secondary folder entries stores a corresponding part of the long file name in Unicode. Windows sets the volume, read-only, system, and hidden file attribute bits of the secondary folder entry to mark it as part of a long file name. MS-DOS and OS/2 generally ignore folder entries with all four of these attribute bits set, so these entries are effectively invisible to these operating systems. Instead, MS-DOS and OS/2 access the file by using the conventional eight-plus-three file name contained in the folder entry for the file.

Figure below shows all of the folder entries for the file Thequi~1.fox, which has a long name of The quick `brown.fox`. The long name is in Unicode, so each character in the name uses two bytes in the folder entry. The attribute field for the long name entries has the value 0x0F. The attribute field for the short name is 0x20.

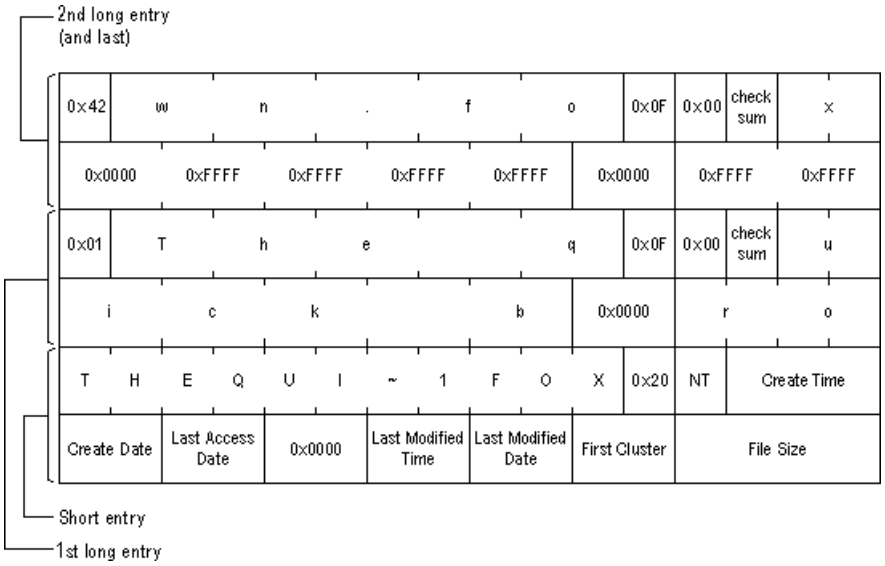


Figure 64: Example of Folder Entries for the long file name

> **Tip:**
>
> For more detailed information see resource kits on Microsoft's web site http://www.microsoft.com/windows/reskits/webresources/default.asp or Microsoft Developers Network (MSDN) http://msdn.microsoft.com

FAT32 Features

Understanding of underlying mechanisms of data storage, organization and data recovery.

FAT32 is a derivative of the File Allocation Table (FAT) file system that supports drives with over 2GB of storage. Because FAT32 drives can contain more than 65,526 clusters, smaller clusters are used than on large FAT16 drives. This method results in more efficient space allocation on the FAT32 drive.

The largest possible file for a FAT32 drive is 4GB minus 2 bytes.

The FAT32 file system includes four bytes per cluster within the file allocation table. Note that the high 4 bits of the 32-bit values in the FAT32 file allocation table are reserved and are not part of the cluster number.

Boot Sector and Bootstrap Modifications

Modifications	Description
Reserved Sectors	FAT32 drives contain more reserved sectors than FAT16 or FAT12 drives. The number of reserved sectors is usually 32, but can vary.
Boot Sector Modifications	Because a FAT32 **BIOS Parameter Block (BPB)**, represented by the BPB structure, is larger than a standard BPB, the boot record on FAT32 drives is greater than 1 sector. In addition, there is a sector in the reserved area on FAT32 drives that contains values for the count of free clusters and the cluster number of the most recently allocated cluster. These values are members of the **BIGFATBOOTFSINFO** structure which is contained within this sector. These additional fields allow the system to initialize the values without having to read the entire file allocation table.
Root Directory	The root directory on a FAT32 drive is not stored in a fixed location as it is on FAT16 and FAT12 drives. On FAT32 drives, the root directory is an ordinary cluster chain. The **A_BF_BPB_RootDirStrtClus** member in the **BPB** structure contains the number of the first cluster in the root directory. This allows the root directory to grow as needed. In addition, the **BPB_RootEntries** member of **BPB** is ignored on a FAT32 drive.
Sectors Per FAT	The **A_BF_BPB_SectorsPerFAT** member of **BPB** is *always* zero on a FAT32 drive. Additionally, the A_BF_BPB_BigSectorsPerFat and **A_BF_BPB_BigSectorsPerFatHi** members of the updated **BPB** provide equivalent information for FAT32 media.

BPB (FAT32)

The BPB for FAT32 drives is an extended version of the FAT16/FAT12 BPB. It contains identical information to a standard BPB, but also includes several extra fields for FAT32 specific information.

This structure is implemented in Windows OEM Service Release 2 and later.

```
A_BF_BPB    STRUC
    A_BF_BPB_BytesPerSector       DW    ?
    A_BF_BPB_SectorsPerCluster    DB    ?
    A_BF_BPB_ReservedSectors      DW    ?
    A_BF_BPB_NumberOfFATs         DB    ?
    A_BF_BPB_RootEntries          DW    ?
    A_BF_BPB_TotalSectors         DW    ?
    A_BF_BPB_MediaDescriptor      DB    ?
    A_BF_BPB_SectorsPerFAT        DW    ?
    A_BF_BPB_SectorsPerTrack      DW    ?
    A_BF_BPB_Heads                DW    ?
    A_BF_BPB_HiddenSectors        DW    ?
    A_BF_BPB_HiddenSectorsHigh    DW    ?
    A_BF_BPB_BigTotalSectors      DW    ?
    A_BF_BPB_BigTotalSectorsHigh  DW    ?
    A_BF_BPB_BigSectorsPerFat     DW    ?
    A_BF_BPB_BigSectorsPerFatHi   DW    ?
    A_BF_BPB_ExtFlags             DW    ?
    A_BF_BPB_FS_Version           DW    ?
    A_BF_BPB_RootDirStrtClus      DW    ?
    A_BF_BPB_RootDirStrtClusHi    DW    ?
    A_BF_BPB_FSInfoSec            DW    ?
    A_BF_BPB_BkUpBootSec          DW    ?
    A_BF_BPB_Reserved             DW    6 DUP (?)
A_BF_BPB    ENDS
```

A_BF_BPB_BytesPerSector

The number of bytes per sector.

A_BF_BPB_SectorsPerCluster

The number of sectors per cluster.

A_BF_BPB_ReservedSectors

The number of reserved sectors, beginning with sector 0.

A_BF_BPB_NumberOfFATs

The number of File Allocation Tables.

A_BF_BPB_RootEntries

This member is ignored on FAT32 drives.

A_BF_BPB_TotalSectors

The size of the partition, in sectors.

A_BF_BPB_MediaDescriptor

The media descriptor. Values in this member are identical to standard BPB.

A_BF_BPB_SectorsPerFAT

The number of sectors per FAT.

Note: This member will always be zero in a FAT32 BPB. Use the values from **A_BF_BPB_BigSectorsPerFat** and **A_BF_BPB_BigSectorsPerFatHi** for FAT32 media.

A_BF_BPB_SectorsPerTrack

The number of sectors per track.

A_BF_BPB_Heads

The number of read/write heads on the drive.

A_BF_BPB_HiddenSectors

The number of hidden sectors on the drive.

A_BF_BPB_HiddenSectorsHigh

The high word of the hidden sectors value.

A_BF_BPB_BigTotalSectors

The total number of sectors on the FAT32 drive.

A_BF_BPB_BigTotalSectorsHigh

The high word of the FAT32 total sectors value.

A_BF_BPB_BigSectorsPerFat

The number of sectors per FAT on the FAT32 drive.

A_BF_BPB_BigSectorsPerFatHi

The high word of the FAT32 sectors per FAT value.

A_BF_BPBExtFlags

Flags describing the drive. Bit 8 of this value indicates whether or not information written to the active FAT will be written to all copies of the FAT. The low 4 bits of this value contain the 0-based FAT number of the Active FAT, but are only meaningful if bit 8 is set. This member can contain a combination of the following values.

Value	Description
BGBPB_F_ActiveFATMask (000Fh)	Mask for low four bits.
BGBPB_F_NoFATMirror (0080h)	Mask indicating FAT mirroring state. If set, FAT mirroring is disabled. If clear, FAT mirroring is enabled.

* Bits 4-6 and 8-15 are reserved.

A_BF_BPB_FS_Version

The file system version number of the FAT32 drive. The high byte represents the major version, and the low byte represents the minor version.

A_BF_BPB_RootDirStrtClus

The cluster number of the first cluster in the FAT32 drive's root directory.

A_BF_BPB_RootDirStrtClusHi

The high word of the FAT32 starting cluster number.

A_BF_BPB_FSInfoSec

The sector number of the file system information sector. The file system info sector contains a **BIGFATBOOTFSINFO** structure. This member is set to 0FFFFh if there is no FSINFO sector. Otherwise, this value must be non-zero and less than the reserved sector count.

A_BF_BPB_BkUpBootSec

The sector number of the backup boot sector. This member is set to 0FFFFh if there is no backup boot sector. Otherwise, this value must be non-zero and less than the reserved sector count.

A_BF_BPB_Reserved

Reserved member.

BIGFATBOOTFSINFO (FAT32)

Contains information about the file system on a FAT32 volume. This structure is implemented in Windows OEM Service Release 2 and later.

```
BIGFATBOOTFSINFO STRUC
  bfFSInf_Sig               DD    ?
  bfFSInf_free_clus_cnt     DD    ?
  bfFSInf_next_free_clus    DD    ?
  bfFSInf_resvd             DD    3 DUP (?)
BIGFATBOOTFSINFO ENDS
```

bfFSInf_Sig

The signature of the file system information sector. The value in this member is FSINFOSIG (0x61417272L).

bfFSInf_free_clus_cnt

The count of free clusters on the drive. Set to -1 when the count is unknown.

bfFSInf_next_free_clus

The cluster number of the cluster that was most recently allocated.

bfFSInf_resvd

Reserved member.

FAT Mirroring

On all FAT drives, there may be multiple copies of the FAT. If an error occurs reading the primary copy, the file system will attempt to read from the backup copies. On FAT16 and FAT12 drives, the first FAT is always the primary copy and any modifications will automatically be written to all copies. However, on FAT32 drives, FAT mirroring can be disabled and a FAT other than the first one can be the primary (or "active") copy of the FAT.

Mirroring is enabled by clearing bit 0x0080 in the **extdpb_flags** member of a FAT32 Drive Parameter Block (DPB) structure, **DPB**.

Mirroring	Description
When Enabled (bit 0x0080 clear)	With mirroring enabled, whenever a FAT sector is written, it will also be written to every other FAT. Also, a mirrored FAT sector can be read from any FAT. A FAT32 drive with multiple FATs will behave the same as FAT16 and FAT12 drives with multiple FATs. That is, the multiple FATs are backups of each other.
When Disabled (bit 0x0080 set)	With mirroring disabled, only one of the FATs is active. The active FAT is the one specified by bits 0 through 3 of the **extdpb_flags** member of **DPB**. The other FATs are ignored. Disabling mirroring allows better handling of a drive with a bad sector in one of the FATs. If a bad sector exists, access to the damaged FAT can be completely disabled. Then, a new FAT can be built in one of the inactive FATs and then made accessible by changing the active FAT value in **extdpb_flags**.

DPB (FAT32)

The DPB was extended to include FAT32 information. Changes are effective for Windows 95 OEM Service Release 2 and later.

```
DPB STRUC
    dpb_drive           DB    ?
    dpb_unit            DB    ?
    dpb_sector_size     DW    ?
    dpb_cluster_mask    DB    ?
    dpb_cluster_shift   DB    ?
    dpb_first_fat       DW    ?
    dpb_fat_count       DB    ?
    dpb_root_entries    DW    ?
    dpb_first_sector    DW    ?
    dpb_max_cluster     DW    ?
    dpb_fat_size        DW    ?
    dpb_dir_sector      DW    ?
    dpb_reserved2       DD    ?
    dpb_media           DB    ?
ifdef NOTFAT32
    dpb_first_access    DB    ?
else
    dpb_reserved        DB    ?
endif
    dpb_reserved3       DD    ?
    dpb_next_free       DW    ?
    dpb_free_cnt        DW    ?
ifndef NOTFAT32
    extdpb_free_cnt_hi  DW    ?
    extdpb_flags        DW    ?
    extdpb_FSInfoSec    DW    ?
    extdpb_BkUpBootSec  DW    ?
    extdpb_first_sector DD    ?
    extdpb_max_cluster  DD    ?
    extdpb_fat_size     DD    ?
    extdpb_root_clus    DD    ?
    extdpb_next_free    DD    ?
endif
DPB ENDS
```

dpb_drive

The drive number (0 = A, 1 = B, and so on).

dpb_unit

Specifies the unit number. The device driver uses the unit number to distinguish the specified drive from the other drives it supports.

dpb_sector_size

The size of each sector, in bytes.

dpb_cluster_mask

The number of sectors per cluster minus 1.

dpb_cluster_shift

The number of sectors per cluster, expressed as a power of 2.

dpb_first_fat

The sector number of the first sector containing the file allocation table (FAT).

dpb_fat_count

The number of FATs on the drive.

dpb_root_entries

The number of entries in the root directory.

dpb_first_sector

The sector number of the first sector in the first cluster.

dpb_max_cluster

The number of clusters on the drive plus 1. This member is undefined for FAT32 drives.

dpb_fat_size

The number of sectors occupied by each FAT. The value of zero indicates a FAT32 drive. Use the value in **extdpb_fat_size** instead.

dpb_dir_sector

The sector number of the first sector containing the root directory. This member is undefined for FAT32 drives.

dpb_reserved2

Reserved member. Do not use.

dpb_media

Specifies the media descriptor for the medium in the specified drive.

reserved

Reserved member. Do not use.

dpb_first_access

Indicates whether the medium in the drive has been accessed. This member is initialized to -1 to force a media check the first time this DPB is used.

dpb_reserved3

Reserved member. Do not use.

dpb_next_free

The cluster number of the most recently allocated cluster.

dpb_free_cnt

The number of free clusters on the medium. This member is 0FFFFh if the number is unknown.

extdpb_free_cnt_hi

The high word of free count.

extdpb_flags

Flags describing the drive. The low 4 bits of this value contain the 0-based FAT number of the Active FAT. This member can contain a combination of the following values.

Value	Description
BGBPB_F_ActiveFATMask (000Fh)	FAT mask for low four bits.
BGBPB_F_NoFATMirror (0080h)	Do not mirror active FAT to inactive FATs.

Bits 4-6 and 8-15 are reserved.

extdpb_FSInfoSec

The sector number of the file system information sector. This member is set to 0FFFFh if there is no FSINFO sector. Otherwise, this value must be non-zero and less than the reserved sector count.

extdpb_BkUpBootSec

The sector number of the backup boot sector. This member is set to 0FFFFh if there is no backup boot sector. Otherwise, this value must be non-zero and less than the reserved sector count.

extdpb_first_sector

The first sector of the first cluster.

extdpb_max_cluster

The number of clusters on the drive plus 1.

extdpb_fat_size

The number of sectors occupied by the FAT.

extdpb_root_clus

The cluster number of the first cluster in the root directory.

extdpb_next_free

The number of the cluster that was most recently allocated.

Partition Types

The following are all the valid partition types and their corresponding values for use in the **Part_FileSystem** member of the **s_partition** structure.

Table 6: Partition Types

Value	Description
PART_UNKNOWN (00h)	Unknown
PART_DOS2_FAT (01h)	12-bit FAT
PART_DOS3_FAT (04h)	16-bit FAT. Partitions smaller than 32MB.
PART_EXTENDED (05h)	Extended MS-DOS Partition
PART_DOS4_FAT (06h)	16-bit FAT. Partitions larger than or equal to 32MB.
PART_DOS32 (0Bh)	32-bit FAT. Partitions up to 2047GB.
PART_DOS32X (0Ch)	Same as PART_DOS32 (0Bh), but uses Logical Block Address Int 13h extensions.
PART_DOSX13 (0Eh)	Same as PART_DOS4_FAT (06h), but uses Logical Block Address Int 13h extensions.
PART_DOSX13X (0Fh)	Same as PART_EXTENDED (05h), but uses Logical Block Address Int 13h extensions.

s_partition (FAT32)

```
s_partition    STRUC
    Part_BootInd        DB    ?
    Part_FirstHead      DB    ?
    Part_FirstSector    DB    ?
    Part_FirstTrack     DB    ?
    Part_FileSystem     DB    ?
    Part_LastHead       DB    ?
    Part_LastSector     DB    ?
    Part_LastTrack      DB    ?
    Part_StartSector    DD    ?
    Part_NumSectors     DD    ?
s_partition    ENDS
```

Part_BootInd

Specifies whether the partition is bootable or not. This value could be set to PART_BOOTABLE (80h), or PART_NON_BOOTABLE(00h). The first partition designated as PART_BOOTABLE is the boot partition. All others are not. Setting multiple partitions to PART_BOOTABLE will result in boot errors.

Part_FirstHead

The first head of this partition. This is a 0-based number representing the offset from the beginning of the disk. The partition includes this head.

Part_FirstSector

The first sector of this partition. This is a 1-based, 6-bit number representing the offset from the beginning of the disk. The partition includes this sector. Bits 0 through 5 specify the 6-bit value; bits 6 and 7 are used with the **Part_FirstTrack** member.

Part_FirstTrack

The first track of this partition. This is an inclusive 0-based, 10-bit number that represents the offset from the beginning of the disk. The high 2 bits of this value are specified by bits 6 and 7 of the **Part_FirstSector** member.

PartFileSystem

Specifies the file system for the partition.

Table 7: Acceptable values

Value	Description
PART_UNKNOWN(00h)	Unknown.
PART_DOS2_FAT(01h)	12-bit FAT.
PART_DOS3_FAT(04h)	16-bit FAT. Partition smaller than 32MB.
PART_EXTENDED(05h)	Extended MS-DOS Partition.
PART_DOS4_FAT(06h)	16-bit FAT. Partition larger than or equal to 32MB.
PART_DOS32(0Bh)	32-bit FAT. Partition up to 2047GB.
PART_DOS32X(0Ch)	Same as PART_DOS32(0Bh), but uses Logical Block Address **Int 13h** extensions.
PART_DOSX13(0Eh)	Same as PART_DOS4_FAT(06h), but uses Logical Block Address **Int 13h** extensions.
PART_DOSX13X(0Fh)	Same as PART_EXTENDED(05h), but uses Logical Block Address **Int 13h** extensions.

Part_LastHead

The last head of the partition. This is a 0-based number that represents the offset from the beginning of the disk. The partition includes the head specified by this member.

Part_LastSector

The last sector of this partition. This is a 1-based, 6-bit number representing offset from the beginning of the disk. The partition includes the sector specified by this member. Bits 0 through 5 specify the 6-bit value; bits 6 and 7 are used with the **Part_LastTrack** member.

Part_LastTrack

The last track of this partition. This is a 0-based, 10-bit number that represents offset from the beginning of the disk. The partition includes this track. The high 2 bits of this value are specified by bits 6 and 7 of the **Part_LastSector** member.

Part_StartSector

Specifies the 1-based number of the first sector on the disk. This value may not be accurate for extended partitions. Use the **Part_FirstSector** value for extended partitions.

Part_NumSectors

The 1-based number of sectors in the partition.

Note:

Values for head and track are 0-based. Sector values are 1-based. This structure is implemented in Windows OEM Service Release 2 and later.

Windows NT File System (NTFS)

Understanding of underlying mechanisms of data storage, organization and data recovery.

The Windows NT file system (NTFS) provides a combination of performance, reliability, and compatibility not found in the FAT file system. It is designed to quickly perform standard file operations such as read, write, and search — and even advanced operations such as file-system recovery — on very large hard disks.

Formatting a volume with the NTFS file system results in the creation of several system files and the Master File Table (MFT), which contains information about all the files and folders on the NTFS volume.

The first information on an NTFS volume is the Partition Boot Sector, which starts at sector 0 and can be up to 16 sectors long. The first file on an NTFS volume is the Master File Table (MFT).

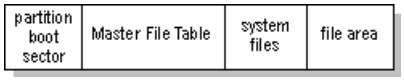


Figure 65: Layout of NTFS volume after formatting

See the next sections for more information about NTFS:

- NTFS Partition Boot Sector on page 156
- NTFS Master File Table (MFT) on page 158
- NTFS file types on page 160
- Data Integrity and Recoverability with NTFS on page 164

The NTFS file system includes security features required for file servers and high-end personal computers in a corporate environment. The NTFS file system also supports data access control and ownership privileges that are important for the integrity of critical data. While folders shared on a Windows NT computer are assigned particular permissions, NTFS files and folders can have permissions assigned whether they are shared or not. NTFS is the only file system on Windows NT that allows you to assign permissions to individual files.

The NTFS file system has a simple, yet very powerful design. Basically, everything on the volume is a file and everything in a file is an attribute, from the data attribute, to the security attribute, to the file name attribute. Every sector on an NTFS volume that is allocated belongs to some file. Even the file system metadata (information that describes the file system itself) is part of a file.

What's New in NTFS5 (Windows 2000)

Encryption
The Encrypting File System (EFS) provides the core file encryption technology used to store encrypted files on NTFS volumes. EFS keeps files safe from intruders who might gain unauthorized physical access to sensitive, stored data (for example, by stealing a portable computer or external disk drive).

Disk quotas
Windows 2000 supports disk quotas for NTFS volumes. You can use disk quotas to monitor and limit disk-space use.

Reparse points
Reparse points are new file system objects in NTFS that can be applied to NTFS files or folders. A file or folder that contains a reparse point acquires additional behaviour not present in the underlying file system. Reparse points are used by many of the new storage features in Windows 2000, including volume mount points.

Volume mount points
Volume mount points are new to NTFS. Based on reparse points, volume mount points allow administrators to graft access to the root of one local volume onto the folder structure of another local volume.

Sparse files
Sparse files allow programs to create very large files but consume disk space only as needed.

Distributed link tracking
NTFS provides a link-tracking service that maintains the integrity of shortcuts to files as well as OLE links within compound documents.

Tip:

For more detailed information see resource kits on Microsoft's web site http://www.microsoft.com/windows/reskits/webresources/default.asp or Microsoft Developers Network (MSDN) http://msdn.microsoft.com

NTFS Partition Boot Sector

Understanding of underlying mechanisms of data storage, organization and data recovery.

Next table describes the boot sector of a volume formatted with NTFS. When you format an NTFS volume, the format program allocates the first 16 sectors for the boot sector and the bootstrap code.

Byte Offset	Field Length	Field Name
0x00	3 bytes	Jump Instruction
0x03	LONGLONG	OEM ID
0x0B	25 bytes	BPB
0x24	48 bytes	Extended BPB
0x54	426 bytes	Bootstrap Code
0x01FE	WORD	End of Sector Marker

On NTFS volumes, the data fields that follow the BPB form an extended BPB. The data in these fields enables Ntldr (NT loader program) to find the master file table (MFT) during start up. On NTFS volumes, the MFT is not located in a predefined sector, as on FAT16 and FAT32 volumes. For this reason, the MFT can be moved if there is a bad sector in its normal location. However, if the data is corrupted, the MFT cannot be located, and Windows NT/2000 assumes that the volume has not been formatted.

The following example illustrates the boot sector of an NTFS volume formatted while running Windows 2000. The printout is formatted in three sections:

- Bytes 0x00– 0x0A are the jump instruction and the OEM ID (shown in bold print).
- Bytes 0x0B–0x53 are the BPB and the extended BPB.
- The remaining code is the bootstrap code and the end of sector marker (shown in bold print).

```
Physical Sector: Cyl 0, Side 1, Sector 1

00000000: EB 52 90 4E 54 46 53 20 - 20 20 20 00 02 08 00 00 .R.NTFS ........
00000010: 00 00 00 00 00 F8 00 00 - 3F 00 FF 00 3F 00 00 00 ........?...?...
00000020: 00 00 00 00 80 00 80 00 - 4A F5 7F 00 00 00 00 00 ........J.......
00000030: 04 00 00 00 00 00 00 00 - 54 FF 07 00 00 00 00 00 ........T.......
00000040: F6 00 00 00 01 00 00 00 - 14 A5 1B 74 C9 1B 74 1C ...........t..t.
00000050: 00 00 00 00 FA 33 C0 8E - D0 BC 00 7C FB B8 C0 07.....3.....|....
00000060: 8E D8 E8 16 00 B8 00 0D - 8E C0 33 DB C6 06 0E 00 ..........3.....
00000070: 10 E8 53 00 68 00 0D 68 - 6A 02 CB 8A 16 24 00 B4 ..S.h..hj....$..
00000080: 08 CD 13 73 05 B9 FF FF - 8A F1 66 0F B6 C6 40 66 ...s......f...@f
00000090: 0F B6 D1 80 E2 3F F7 E2 - 86 CD C0 ED 06 41 66 0F .....?.......Af.
000000A0: B7 C9 66 F7 E1 66 A3 20 - 00 C3 B4 41 BB AA 55 8A ..f..f. ...A..U.
000000B0: 16 24 00 CD 13 72 0F 81 - FB 55 AA 75 09 F6 C1 01 .$...r...U.u....
000000C0: 74 04 FE 06 14 00 C3 66 - 60 1E 06 66 A1 10 00 66 t......f`..f...f
000000D0: 03 06 1C 00 66 3B 06 20 - 00 0F 82 3A 00 1E 66 6A ....f;. ...:..fj
000000E0: 00 66 50 06 53 66 68 10 - 00 01 00 80 3E 14 00 00 .fP.Sfh.....>...
000000F0: 0F 85 0C 00 E8 B3 FF 80 - 3E 14 00 00 0F 84 61 00 ........>.....a.
00000100: B4 42 8A 16 24 00 16 1F - 8B F4 CD 13 66 58 5B 07 .B..$.......fX[..
00000110: 66 58 66 58 1F EB 2D 66 - 33 D2 66 0F B7 0E 18 00 fXfX.-f3.f......
00000120: 66 F7 F1 FE C2 8A CA 66 - 8B D0 66 C1 EA 10 F7 36 f......f..f....6
00000130: 1A 00 86 D6 8A 16 24 00 - 8A E8 C0 E4 06 0A CC B8 ......$.........
00000140: 01 02 CD 13 0F 82 19 00 - 8C C0 05 20 00 8E C0 66 ........... ...f
00000150: FF 06 10 00 FF 0E 0E 00 - 0F 85 6F FF 07 1F 66 61 ..........o...fa
00000160: C3 A0 F8 01 E8 09 00 A0 - FB 01 E8 03 00 FB EB FE ................
00000170: B4 01 8B F0 AC 3C 00 74 - 09 B4 0E BB 07 00 CD 10 .....<.t........
00000180: EB F2 C3 0D 0A 41 20 64 - 69 73 6B 20 72 65 61 64 .....A disk read
00000190: 20 65 72 72 6F 72 20 6F - 63 63 75 72 72 65 64 00  error occurred.
000001A0: 0D 0A 4E 54 4C 44 52 20 - 69 73 20 6D 69 73 73 69 ..NTLDR is missi
000001B0: 6E 67 00 0D 0A 4E 54 4C - 44 52 20 69 73 20 63 6F ng...NTLDR is co
000001C0: 6D 70 72 65 73 73 65 64 - 00 0D 0A 50 72 65 73 73 mpressed...Press
000001D0: 20 43 74 72 6C 2B 41 6C - 74 2B 44 65 6C 20 74 6F  Ctrl+Alt+Del to
000001E0: 20 72 65 73 74 61 72 74 - 0D 0A 00 00 00 00 00 00  restart........
000001F0: 00 00 00 00 00 00 00 00 - 83 A0 B3 C9 00 00 55 AA ..............U.
```

The following table describes the fields in the BPB and the extended BPB on NTFS volumes. The fields starting at 0x0B, 0x0D, 0x15, 0x18, 0x1A, and 0x1C match those on FAT16 and FAT32 volumes. The sample values correspond to the data in this example.

Table 8: BIOS Parameter Block and Extended BIOS Parameter Block Fields

Byte Offset	Field Length	Sample Value	Field Name
0x0B	WORD	0x0002	Bytes Per Sector
0x0D	BYTE	0x08	Sectors Per Cluster
0x0E	WORD	0x0000	Reserved Sectors
0x10	3 BYTES	0x000000	*always 0*
0x13	WORD	0x0000	*not used by NTFS*
0x15	BYTE	0xF8	Media Descriptor
0x16	WORD	0x0000	*always 0*
0x18	WORD	0x3F00	Sectors Per Track

Byte Offset	Field Length	Sample Value	Field Name
0x1A	WORD	0xFF00	Number Of Heads
0x1C	DWORD	0x3F000000	Hidden Sectors
0x20	DWORD	0x00000000	*Unused by NTFS*
0x24	DWORD	0x80008000	*Unused by NTFS*
0x28	LONGLONG	0x4AF576015E06000	Total Sectors
0x30	LONGLONG	0x0400000000000000	Logical Cluster Number for the file $MFT
0x38	LONGLONG	0x54FF0700000000000	Logical Cluster Number for the file $MFTMirr
0x40	DWORD	0xF6000000	Clusters Per File Record Segment
0x44	DWORD	0x01000000	Clusters Per Index Block
0x48	LONGLONG	0x14A5787A4C91B741	Volume Serial Number
0x50	DWORD	0x00000000	Checksum

Protecting the Boot Sector

Because a normally functioning system relies on the boot sector to access a volume, it is highly recommended that you run disk scanning tools such as **chkdsk** regularly, as well as back up all of your data files to protect against data loss if you lose access to a volume.

Tip:

For more detailed information see resource kits on Microsoft's web site http://www.microsoft.com/windows/reskits/webresources/default.asp or Microsoft Developers Network (MSDN) http://msdn.microsoft.com

NTFS Master File Table (MFT)

Understanding of underlying mechanisms of data storage, organization and data recovery.

Each file on an NTFS volume is represented by a record in a special file called the master file table (MFT). NTFS reserves the first 16 records of the table for special information. The first record of this table describes the master file table itself, followed by a MFT *mirror record*. If the first MFT record is corrupted, NTFS reads the second record to find the MFT mirror file, whose first record is identical to the first record of the MFT. The locations of the data segments for both the MFT and MFT mirror file are recorded in the boot sector. A duplicate of the boot sector is located at the logical center of the disk.

The third record of the MFT is the log file, used for file recovery. The seventeenth and following records of the master file table are for each file and directory (also viewed as a file by NTFS) on the volume.

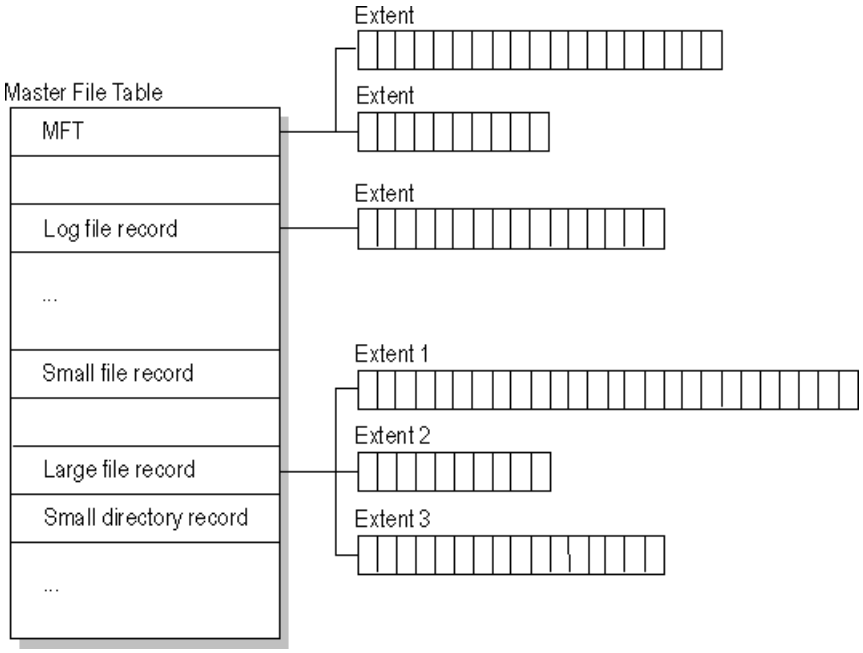


Figure 66: Simplified illustration of the MFT structure

The master file table allocates a certain amount of space for each file record. The attributes of a file are written to the allocated space in the MFT. Small files and directories (typically 1500 bytes or smaller), such as the file illustrated in next figure, can entirely be contained within the master file table record.

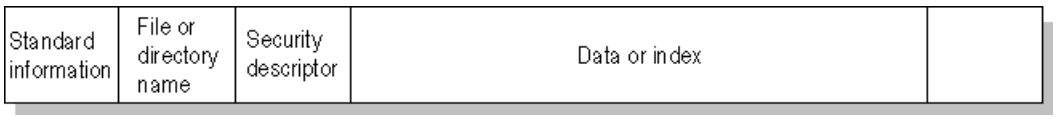


Figure 67: MFT Record for a Small File or Directory

This design makes file access very fast. Consider, for example, the FAT file system, which uses a file allocation table to list the names and addresses of each file. FAT directory entries contain an index into the file allocation table. When you want to view a file, FAT first reads the file allocation table and assures that it exists. Then FAT retrieves the file by searching the chain of allocation units assigned to the file. With NTFS, as soon as you look up the file, it's there for you to use.

Directory records are housed within the master file table just like file records. Instead of data, directories contain index information. Small directory records reside entirely within the MFT structure. Large directories are organized into B-trees, having records with pointers to external clusters containing directory entries that could not be contained within the MFT structure.

Tip:

For more detailed information see resource kits on Microsoft's web site http://www.microsoft.com/windows/reskits/webresources/default.asp or Microsoft Developers Network (MSDN) http://msdn.microsoft.com

NTFS file types
Understanding of underlying mechanisms of data storage, organization and data recovery.

NTFS file attributes

The NTFS file system views each file (or folder) as a set of file attributes. Elements such as the file's name, its security information, and even its data, are all file attributes. Each attribute is identified by an attribute type code and, optionally, an attribute name.

When a file's attributes can fit within the MFT file record, they are called resident attributes. For example, information such as file name and time stamp are always included in the MFT file record. When all of the information for a file is too large to fit in the MFT file record, some of its attributes are non-resident. The non-resident attributes are allocated one or more clusters of disk space elsewhere in the volume. NTFS creates the Attribute List attribute to describe the location of all of the attribute records.

Next table lists all of the file attributes currently defined by the NTFS file system. This list is extensible, meaning that other file attributes can be defined in the future.

Attribute Type	Description
Standard Information	Includes information such as timestamp and link count.
Attribute List	Lists the location of all attribute records that do not fit in the MFT record.
File Name	A repeatable attribute for both long and short file names. The long name of the file can be up to 255 Unicode characters. The short name is the 8.3, case-insensitive name for the file. Additional names, or hard links, required by POSIX can be included as additional file name attributes.
Security Descriptor	Describes who owns the file and who can access it.
Data	Contains file data. NTFS allows multiple data attributes per file. Each file typically has one unnamed data attribute. A file can also have one or more named data attributes, each using a particular syntax.
Object ID	A volume-unique file identifier. Used by the distributed link tracking service. Not all files have object identifiers.
Logged Tool Stream	Similar to a data stream, but operations are logged to the NTFS log file just like NTFS metadata changes. This is used by EFS.
Reparse Point	Used for volume mount points. They are also used by Installable File System (IFS) filter drivers to mark certain files as special to that driver.
Index Root	Used to implement folders and other indexes.
Index Allocation	Used to implement folders and other indexes.
Bitmap	Used to implement folders and other indexes.
Volume Information	Used only in the $Volume system file. Contains the volume version.
Volume Name	Used only in the $Volume system file. Contains the volume label.

NTFS system files

NTFS includes several system files, all of which are hidden from view on the NTFS volume. A *system file* is one used by the file system to store its meta data and to implement the file system. System files are placed on the volume by the Format utility.

Table 9: Meta-data Stored in the Master File Table

System File	File Name	MFT Record	Purpose of the File
Master file table	$Mft	0	Contains one base file record for each file and folder on an NTFS volume. If the allocation information for a file or folder is too large to fit within a single record, other file records are allocated as well.
Master file table 2	$MftMirr	1	A duplicate image of the first four records of the MFT. This file guarantees access to the MFT in case of a single-sector failure.
Log file	$LogFile	2	Contains a list of transaction steps used for NTFS recoverability. Log file size depends on the volume size and can be as large as 4 MB. It is used by Windows NT/2000 to restore consistency to NTFS after a system failure.
Volume	$Volume	3	Contains information about the volume, such as the volume label and the volume version.
Attribute definitions	$AttrDef	4	A table of attribute names, numbers, and descriptions.
Root file name index	$	5	The root folder.
Cluster bitmap	$Bitmap	6	A representation of the volume showing which clusters are in use.
Boot sector	$Boot	7	Includes the BPB used to mount the volume and additional bootstrap loader code used if the volume is bootable.
Bad cluster file	$BadClus	8	Contains bad clusters for the volume.
Security file	$Secure	9	Contains unique security descriptors for all files within a volume.
Upcase table	$Upcase	10	Converts lowercase characters to matching Unicode uppercase characters.
NTFS extension file	$Extend	11	Used for various optional extensions such as quotas, reparse point data, and object identifiers.
		12–15	Reserved for future use.

NTFS multiple data streams

NTFS supports multiple data streams, where the stream name identifies a new data attribute on the file. A handle can be opened to each data stream. A data stream, then, is a unique set of file attributes. Streams have separate opportunistic locks, file locks, and sizes, but common permissions.

This feature enables you to manage data as a single unit. The following is an example of an alternate stream:

```
myfile.dat:stream2
```

A library of files might exist where the files are defined as alternate streams, as in the following example:

```
library:file1
                :file2
                :file3
```

A file can be associated with more than one application at a time, such as Microsoft™ Word and Microsoft™ WordPad. For instance, a file structure like the following illustrates file association, but not multiple files:

```
program:source_file
```

```
       :doc_file
```

```
       :object_file
```

```
       :executable_file
```

To create an alternate data stream, at the command prompt, you can type commands such as:

```
echo text>program:source_file
```

```
more <program:source_file
```

Important:

When you copy an NTFS file to a FAT volume, such as a floppy disk, data streams and other attributes not supported by FAT are lost.

NTFS compressed files

Windows NT/2000 supports compression on individual files, folders, and entire NTFS volumes. Files compressed on an NTFS volume can be read and written by any Windows-based application without first being decompressed by another program. Decompression occurs automatically when the file is read. The file is compressed again when it is closed or saved. Compressed files and folders have an attribute of **C** when viewed in Windows Explorer.

Only NTFS can read the compressed form of the data. When an application such as Microsoft™ Word or an operating system command such as **copy** requests access to the file, the compression filter driver decompresses the file before making it available. For example, if you copy a compressed file from another Windows NT/2000–based computer to a compressed folder on your hard disk, the file is decompressed when read, copied, and then re-compressed when saved.

This compression algorithm is similar to that used by the Windows 98 application DriveSpace 3, with one important difference — the limited functionality compresses the entire primary volume or logical volume. NTFS allows for the compression of an entire volume, of one or more folders within a volume, or even one or more files within a folder of an NTFS volume.

The compression algorithms in NTFS are designed to support cluster sizes of up to 4 KB. When the cluster size is greater than 4 KB on an NTFS volume, none of the NTFS compression functions are available.

Each NTFS data stream contains information that indicates whether any part of the stream is compressed. Individual compressed buffers are identified by “holes” following them in the information stored for that stream. If there is a hole, NTFS automatically decompresses the preceding buffer to fill the hole.

NTFS provides real-time access to a compressed file, decompressing the file when it is opened and compressing it when it is closed. When writing a compressed file, the system reserves disk space for the uncompressed size. The system gets back unused space as each individual compression buffer is compressed.

NTFS encrypted files (Windows 2000 only)

The Encrypting File System (EFS) provides the core file encryption technology used to store encrypted files on NTFS volumes. EFS keeps files safe from intruders who might gain unauthorized physical access to sensitive, stored data (for example, by stealing a portable computer or external disk drive).

EFS uses symmetric key encryption in conjunction with public key technology to protect files and ensure that only the owner of a file can access it. Users of EFS are issued a digital certificate with a public key and a private key pair. EFS uses the key set for the user who is logged on to the local computer where the private key is stored.

Users work with encrypted files and folders just as they do with any other files and folders. Encryption is transparent to the user who encrypted the file; the system automatically decrypts the file or folder when the user accesses. When the file is saved, encryption is reapplied. However, intruders who try to access the encrypted files or folders receive an "Access denied" message if they try to open, copy, move, or rename the encrypted file or folder.

To encrypt or decrypt a folder or file, set the encryption attribute for folders and files just as you set any other attribute. If you encrypt a folder, all files and subfolders created in the encrypted folder are automatically encrypted. It is recommended that you encrypt at the folder level.

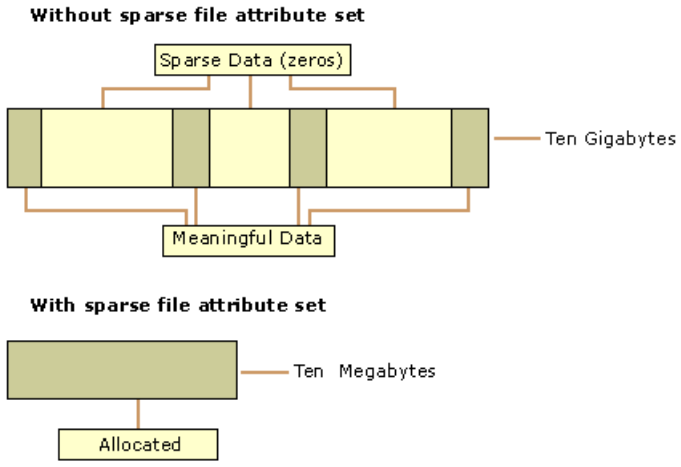
NTFS sparse files (Windows 2000 only)

A sparse file has an attribute that causes the I/O subsystem to allocate only meaningful (nonzero) data. Nonzero data is allocated on disk, and non-meaningful data (large strings of data composed of zeros) is not. When a sparse file is read, allocated data is returned as it was stored; non-allocated data is returned, by default, as zeros.

NTFS deallocates sparse data streams and only maintains other data as allocated. When a program accesses a sparse file, the file system yields allocated data as actual data and deallocated data as zeros.

NTFS includes full sparse file support for both compressed and uncompressed files. NTFS handles read operations on sparse files by returning allocated data and sparse data. It is possible to read a sparse file as allocated data and a range of data without retrieving the entire data set, although NTFS returns the entire data set by default.

With the sparse file attribute set, the file system can deallocate data from anywhere in the file and, when an application calls, yield the zero data by range instead of storing and returning the actual data. File system application programming interfaces (APIs) allow for the file to be copied or backed as actual bits and sparse stream ranges. The net result is efficient file system storage and access. Next figure shows how data is stored with and without the sparse file attribute set.



Important:

If you copy or move a sparse file to a FAT or a non-Windows 2000 NTFS volume, the file is built to its originally specified size. If the required space is not available, the operation does not complete.

Tip:

For more detailed information see resource kits on Microsoft's web site http://www.microsoft.com/windows/reskits/webresources/default.asp or Microsoft Developers Network (MSDN) http://msdn.microsoft.com

Data Integrity and Recoverability with NTFS
Understanding of underlying mechanisms of data storage, organization and data recovery.

NTFS is a recoverable file system that guarantees the consistency of the volume by using standard transaction logging and recovery techniques. In the event of a disk failure, NTFS restores consistency by running a recovery procedure that accesses information stored in a log file. The NTFS recovery procedure is exact, guaranteeing that the volume is restored to a consistent state. Transaction logging requires a very small amount of overhead.

NTFS ensures the integrity of all NTFS volumes by automatically performing disk recovery operations the first time a program accesses an NTFS volume after the computer is restarted following a failure.

NTFS also uses a technique called cluster remapping to minimize the effects of a bad sector on an NTFS volume.

Important:

If either the master boot record (MBR) or boot sector is corrupted, you might not be able to access data on the volume.

Recovering Data with NTFS

NTFS views each I/O operation that modifies a system file on the NTFS volume as a transaction, and manages each one as an integral unit. Once started, the transaction is either completed or, in the event of a disk failure, rolled back (such as when the NTFS volume is returned to the state it was in before the transaction was initiated).

To ensure that a transaction can be completed or rolled back, NTFS records the suboperations of a transaction in a log file before they are written to the disk. When a complete transaction is recorded in the log file, NTFS performs the suboperations of the transaction on the volume cache. After NTFS updates the cache, it commits the transaction by recording in the log file that the entire transaction is complete.

Once a transaction is committed, NTFS ensures that the entire transaction appears on the volume, even if the disk fails. During recovery operations, NTFS redoes each committed transaction found in the log file. Then NTFS locates the transactions in the log file that were not committed at the time of the system failure and undoes each transaction suboperation recorded in the log file. Incomplete modifications to the volume are prohibited.

NTFS uses the Log File service to log all redo and undo information for a transaction. NTFS uses the redo information to repeat the transaction. The undo information enables NTFS to undo transactions that are not complete or that have an error.

Important:

NTFS uses transaction logging and recovery to guarantee that the volume structure is not corrupted. For this reason, all system files remain accessible after a system failure. However, user data can be lost because of a system failure or a bad sector.

Cluster Remapping

In the event of a bad-sector error, NTFS implements a recovery technique called cluster remapping. When Windows 2000 detects a bad-sector, NTFS dynamically remaps the cluster containing the bad sector and allocates a new cluster for the data. If the error occurred during a read, NTFS returns a read error to the calling program, and the data is lost. If the error occurs during a write, NTFS writes the data to the new cluster, and no data is lost.

NTFS puts the address of the cluster containing the bad sector in its bad cluster file so the bad sector is not reused.

Important:

Cluster remapping is *not* a backup alternative. Once errors are detected, the disk should be monitored closely and replaced if the defect list grows. This type of error is displayed in the Event Log.

Tip:

For more detailed information see resource kits on Microsoft's web site http://www.microsoft.com/windows/reskits/webresources/default.asp or Microsoft Developers Network (MSDN) http://msdn.microsoft.com

Hierarchical File System (HFS)

Understanding of underlying mechanisms of data storage, organization and data recovery.

Hierarchical File System (HFS) is a proprietary file system developed by Apple Inc. for use in computer systems running Mac OS. Originally designed for use on floppy and hard disks, it can also be found on read-only media such as CD-ROMs. HFS is also referred to as Mac OS Standard (or HFS Standard), while its successor, HFS Plus, is also called Mac OS Extended (or HFS Extended). With the introduction of Mac OS X 10.6, Apple dropped support for formatting or writing HFS Standard disks and images, which remained supported as read-only volumes until macOS 10.15. Starting with macOS 10.15, HFS Standard disks can no longer be read.

A storage volume is inherently divided into logical blocks of 512 bytes. The Hierarchical File System groups these logical blocks into allocation blocks, which can contain one or more logical blocks, depending on the total size of the volume. HFS uses a 16-bit value to address allocation blocks, limiting the number of allocation blocks to 65,535 (2^16-1).

Five structures make up an HFS volume:

1. Logical blocks 0 and 1 of the volume are the **Boot Blocks**, which contain system startup information. For example, the names of the System and Shell (usually the Finder) files which are loaded at startup.
2. Logical block 2 contains the **Master Directory Block** (also known as **MDB**). This defines a wide variety of data about the volume itself, for example date and time stamps for when the volume was created, the location of the other volume structures such as the Volume Bitmap or the size of logical structures such as allocation blocks. There is also a duplicate of the MDB called the **Alternate Master Directory Block** (also known as **Alternate MDB**) located at the opposite end of the volume in the second to last logical block. This is intended mainly for use by disk utilities and is only updated when either the Catalog File or Extents Overflow File grow in size.
3. Logical block 3 is the starting block of the **Volume Bitmap**, which keeps track of which allocation blocks are in use and which are free. Each allocation block on the volume is represented by a bit in the map: if the bit is set then the block is in use; if it is clear then the block is free to be used. Since the Volume Bitmap must have a bit to represent each allocation block, its size is determined by the size of the volume itself.
4. The **Extent Overflow File** is a B-tree that contains extra extents that record which allocation blocks are allocated to which files, once the initial three extents in the Catalog File are used up. Later versions also

added the ability for the Extent Overflow File to store extents that record bad blocks, to prevent the file system from trying to allocate a bad block to a file.

5. The **Catalog File** is another B-tree that contains records for all the files and directories stored in the volume. It stores four types of records. Each file consists of a File Thread Record and a File Record while each directory consists of a Directory Thread Record and a Directory Record. Files and directories in the Catalog File are located by their unique **Catalog Node ID** (or **CNID**).
   - A **File Thread Record** stores just the name of the file and the CNID of its parent directory.
   - A **File Record** stores a variety of metadata about the file including its CNID, the size of the file, three timestamps (when the file was created, last modified, last backed up), the first file extents of the data and resource forks and pointers to the file's first data and resource extent records in the Extent Overflow File. The File Record also stores two 16 byte fields that are used by the Finder to store attributes about the file including things like its creator code, type code, the window the file should appear in and its location within the window.
   - A **Directory Thread Recor**d stores just the name of the directory and the CNID of its parent directory.
   - A **Directory Record** which stores data like the number of files stored within the directory, the CNID of the directory, three timestamps (when the directory was created, last modified, last backed up). Like the File Record, the Directory Record also stores two 16 byte fields for use by the Finder. These store things like the width and height and x and y co-ordinates for the window used to display the contents of the directory, the display mode (icon view, list view, etc.) of the window and the position of the window's scroll bar.

Hierarchical File System Plus (HFS+)

HFS Plus volumes are divided into sectors (called logical blocks in HFS), that are usually 512 bytes in size. These sectors are then grouped together into allocation blocks which can contain one or more sectors; the number of allocation blocks depends on the total size of the volume. HFS Plus uses a larger value to address allocation blocks than HFS, 32 bits rather than 16 bits; this means it can access 4,294,967,296 (= 232) allocation blocks rather than the 65,536 (= 216) allocation blocks available to HFS.[2] When disks were small, this was of little consequence, but as larger-capacity drives became available, it meant that the smallest amount of space that any file could occupy (a single allocation block) became excessively large, wasting significant amounts of space. For example, on a 1 GB disk, the allocation block size under HFS is 16 KB, so even a 1-byte file would take up 16 KB of disk space. HFS Plus's system greatly improves space utilization on larger disks as a result.

File and folder names in HFS Plus are also encoded in UTF-16[13] and normalized to a form very nearly the same as Unicode Normalization Form D (NFD)[14] (which means that precomposed characters like "å" are decomposed in the HFS+ filename and therefore count as two code units[15] and UTF-16 implies that characters from outside the Basic Multilingual Plane also count as two code units in an HFS+ filename). HFS Plus permits filenames up to 255 UTF-16 code units in length.

Formerly, HFS Plus volumes were embedded inside an HFS standard file system. This was phased out by the Tiger transition to Intel Macs, where the HFS Plus file system was not embedded inside a wrapper. The wrapper had been designed for two purposes; it allowed Macintosh computers without HFS Plus support in their ROM to boot HFS Plus volumes and it also was designed to help users transition to HFS Plus by including a minimal HFS volume with a read-only file called Where_have_all_my_files_gone?, explaining to users with versions of Mac OS 8.0 and earlier without HFS Plus, that the volume requires a system with HFS Plus support. The original HFS volume contains a signature and an offset to the embedded HFS Plus volume within its volume header. All allocation blocks in the HFS volume which contain the embedded volume are mapped out of the HFS allocation file as bad blocks.

Notable among file systems used for Unix systems, HFS Plus does not support sparse files.

There are nine structures that make up a typical HFS Plus volume:

1. Sectors 0 and 1 of the volume are HFS **boot blocks**. These are identical to the boot blocks in an HFS volume. They are part of the HFS wrapper.

2. Sector 2 contains the **Volume Header**, which is equivalent to the Master Directory Block in an HFS volume. The Volume Header stores a wide variety of data about the volume itself, for example the size of allocation blocks, a timestamp that indicates when the volume was created or the location of other volume structures such as the Catalog File or Extent Overflow File. The Volume Header is always located in the same place.
3. The **Allocation File** which keeps track of which allocation blocks are free and which are in use. It is similar to the Volume Bitmap in HFS, in which each allocation block is represented by one bit. A zero means the block is free and a one means the block is in use. The main difference with the HFS Volume Bitmap, is that the Allocation File is stored as a regular file – it does not occupy a special reserved space near the beginning of the volume. The Allocation File can also change size and does not have to be stored contiguously within a volume.
4. The **Catalog File** is a B-tree that contains records for all the files and directories stored in the volume. The HFS Plus Catalog File is very similar to the HFS Catalog File, the main differences being records are larger to allow more fields and to allow for those fields to be larger (for example to allow the longer 255-character unicode file names in HFS Plus). A record in the HFS Catalog File is 512 bytes in size; a record in the HFS Plus Catalog File is 4 KB in the classic Mac OS and 8 KB in macOS. Fields in HFS are of fixed size, while in HFS Plus the size can vary depending on the actual size of the data they store.
5. The **Extents Overflow File** is another B-tree that records the allocation blocks that are allocated to each file as extents. Each file record in the Catalog File is capable of recording eight extents for each fork of a file; once those are used additional extents are recorded in the Extents Overflow File. Bad blocks are also recorded as extents in the Extents Overflow File. The default size of an extent record in the classic Mac OS is 1 KB and 4 KB in macOS.
6. The **Attributes File** is a new B-tree in HFS Plus that does not have a corresponding structure in HFS. The Attributes File can store three different types of 4 KB records: Inline Data Attribute records, Fork Data Attribute records and Extension Attribute records. Inline Data Attribute records store small attributes that can fit within the record itself. Fork Data Attribute records contain references to a maximum of eight extents that can hold larger attributes. Extension Attributes are used to extend a Fork Data Attribute record when its eight extent records are already used.
7. The **Startup File** is designed for non-Mac OS systems that lack HFS or HFS Plus support. It is similar to the Boot Blocks of an HFS volume.
8. The second-to-last sector contains the **Alternate Volume Header**, which is equivalent to the Alternate Master Directory Block of HFS. This is the second-to-last-sector for the disk, not the volume; if the disk is larger than the volume, the AVH will be outside the range of the filesystem.
9. The last sector in the volume is reserved for use by Apple. It is used during the computer manufacturing process.

Apple File System (ApFS)

ApFS is proprietary file system created by Apple to replace HFS+ as the primary storage architecture across its devices. Designed with modern storage needs in mind, APFS offers enhanced performance, security, and reliability. It supports features such as strong native encryption for data protection, space sharing for efficient storage management, and instant snapshots for quick backups and restore points. Additionally, APFS is optimized for *Solid-state drive (SSD)* and flash storage, providing faster data access and improved responsiveness. Its advanced architecture ensures data integrity through crash protection and seamless scalability, making it suitable for a wide range of Apple devices, from smartphones and tablets to laptops and desktops.

just term: *Solid-state drive (SSD)*

Extended File System (exFAT)

Understanding of underlying mechanisms of data storage, organization and data recovery.

Extended File System (exFAT) is a successor of FAT family of file systems (FAT12/16/32). It has similar design though renders many significant improvements:

- Larger volume and file size limits
- Native Unicode file names

- Bigger boot area allowing a larger boot code
- Better performance
- Time zone offset support
- OEM parameters support

Table 10: exFAT vs. FAT32 Comparison

Feature	FAT32	exFAT
Maximum Volume Size	8 TB*	128 PB
Maximum File Size	4 GB	16 EB
Maximum Cluster Size	32 KB **	32 MB
Maximum Cluster Count	228	232
Maximum File Name Length	255	255
Date/Time resolution	2 s	10 ms
MBR Partition Type Identifier	0x0B, 0x0C	0x07

Notice: Windows cannot format FAT32 volumes bigger than 32GB, though it supports larger volumes created by third party implementations; 16 TB is the maximum volume size if formatted with 64KB cluster

Notice: According to Microsoft KB184006 clusters cannot be 64KB or larger, though some third party implementations support up to 64KB.

Related concepts

Volume Layout on page 168
Understanding of underlying mechanisms of data storage, organization and data recovery.

exFAT Directory Structure on page 173
Understanding of underlying mechanisms of data storage, organization and data recovery.

exFAT Defined Directory Entries on page 175
Understanding of underlying mechanisms of data storage, organization and data recovery.

exFAT Cluster Heap on page 182
Understanding of underlying mechanisms of data storage, organization and data recovery.

Volume Layout

Understanding of underlying mechanisms of data storage, organization and data recovery.

Offset, sectors	Size, sectors	Block	Comments
Main Boot Region			
0	1	Boot Sector	
1	8	Extended Boot Sectors	
9	1	OEM Parameters	
10	1	Reserved	
11	1	Boot Checksum	
Backup Boot Region			
12	1	Boot Sector	

Offset, sectors	Size, sectors	Block	Comments
13	8	Extended Boot Sectors	
21	1	OEM Parameters	
22	1	Reserved	
23	1	Boot Checksum	
FAT Region			
24	FatOffset - 24	FAT Alignment	Boot Sectors contain FatOffset
FatOffset	FatLength	First FAT	Boot Sectors contain FatOffset and FatLength
FatOffset + FatLength	FatLength	Second FAT	For TexFAT only
Data Region			
FatOffset + FatLength * NumberOfFats	ClusterHeapOffset – (FatOffset + FatLength * NumberOfFats)	Cluster Heap Alignment	
ClusterHeapOffset	ClusterCount * 2^SectorsPerClusterShift	Cluster Heap	
ClusterHeapOffset + ClusterCount * 2^SectorsPerClusterShift	VolumeLength – (ClusterHeapOffset + ClusterCount * 2^SectorsPerClusterShift)	Excess Space	

Navigate to detailed volume specification using following links:

- Boot Sector on page 169
- Extended Boot Sector on page 171
- OEM Parameters on page 171
- File Allocation Table (FAT) on page 172

Boot Sector

Offset	Size	Description	Comments
0 (0x00)	3	JumpBoot	0xEB7690
3 (0x03)	8	FileSystemName	"EXFAT "
11 (0x0B)	53	MustBeZero	
64 (0x40)	8	PartitionOffset	In sectors; if 0, shall be ignored
72 (0x48)	8	VolumeLength	Size of exFAT volume in sectors
80 (0x50)	4	FatOffset	In sectors
84 (0x54)	4	FatLength	In sectors. May exceed the required space in order to align the second FAT
88 (0x58)	4	ClusterHeapOffset	In sectors

Offset	Size	Description	Comments
92 (0x5C)	4	ClusterCount	2^32-11 is the maximum number of clusters could be described
96 (0x60)	4	RootDirectoryCluster	
100 (0x64)	4	VolumeSerialNumber	
104 (0x68)	2	FileSystemRevision	as MAJOR.minor, major revision is high byte, minor is low byte; currently 01.00
106 (0x6A)	2	VolumeFlags (see below)	
108 (0x6C)	1	BytesPerSectorShift	Power of 2. Minimum 9 (512 bytes per sector), maximum 12 (4096 bytes per sector)
109 (0x6D)	1	SectorsPerCluster Shift	Power of 2. Minimum 0 (1 sector per cluster), maximum 25 – BytesPerSectorShift, so max cluster size is 32 MB
110 (0x6E)	1	NumberOfFats	2 is for TexFAT only
111 (0x6F)	1	DriveSelect	Extended INT 13h drive number; typically 0x80
112 (0x70)	1	PercentInUse	0..100 – percentage of allocated clusters rounded down to the integer 0xFF – percentage is not available
113 (0x71)	7	Reserved	
120 (0x78)	390	BootCode	
510 (0x1FE)	2	BootSignature	0xAA55
512 (0x200)	2^BytesPerSectorShift - 512	ExcessSpace	Not used

Table 11: Volume Flags

Offset	Size	Field
0	1	ActiveFat 0 - First FAT and Allocation Bitmap are active, 1 - Second .
1	1	VolumeDirty (0-clean, 1-dirty)
2	1	MediaFailure (0 – no failures reported or they already marked as BAD clusters) 1- some read/ write operations failed)

Offset	Size	Field
3	1	ClearToZero (no meaning)
4	12	Reserved

Extended Boot Sector

Offset	Size	Description	Comments
0 (0x00)	2^BytesPerSectorShift - 4	ExtendedBootCode	
2^BytesPerSectorShift - 4	4	ExtendedBootSignature	0xAA550000

Whole sector is used for boot code except last 4 bytes used for signature in each sector. If Extended Boot Sector is not used, it should be filled with 0x00. Extended signature must be preserved.

OEM Parameters

Offset	Size	Description	Comments
0 (0x00)	48	Parameters[0]	
...	...	...	
432 (0x1B0)	48	Parameters[9]	
480 (0x01E0)	2^BytesPerSectorShift - 480	Reserved	

OEM parameters are ignored by Windows but can be used by OEM implementations. OEMs can define their own parameters with unique GUIDs. All unused Parameters fields must be described as unused by GUID_NULL in ParameterType.

This structure must be preserved during exFAT formatting, except in the case of secure wipe.

Table 12: OEM Parameter Record

Offset	Size	Description	Comments
0x00	16	ParameterType	OEM defined GUID , GUID_NULL indicate that parameter value is not used
0x10	32	ParameterValue	OEM specific

```
#define OEM_FLASH_PARAMETER_GUID 0A0C7E46-3399-4021-90C8-FA6D389C4BA2
struct
{
 GUID OemParameterType; //Value is OEM_FLASH_PARAMETER_GUID
 UINT32 EraseBlockSize; //Erase block size in bytes
 UINT32 PageSize;
 UINT32 NumberOfSpareBlocks;
 UINT32 tRandomAccess;  //Random Access Time in nanoseconds
 UINT32 tProgram;            //Program time in nanoseconds
 UINT32 tReadCycle;     //Serial read cycle time in nanoseconds
 UINT32 tWriteCycle;    //Write Cycle time in nanoseconds
 UCHAR Reserved[4];
}
FlashParameters;
```

Boot Checksum

This sector contains a repeating 32-bit checksum of the previous 11 sectors. The checksum calculation excludes VolumeFlags and PercentInUse fields in Boot Sector (bytes 106, 107, 112). The checksum is repeated until the end of the sector. The number of repetitions depends on the size of the sector.

```
UINT32 BootChecksum(const unsigned char data[], int bytes)
{
 UINT32 checksum = 0;

 for (int i = 0; i < bytes; i++)
 {
  if (i == 106 || i == 107 || i == 112)
   continue;
  checksum = (checksum << 31) | (checksum >> 1) + data[i];
 }
 return checksum;
}
```

File Allocation Table (FAT)

File Allocation Table (FAT) may contain 1 or 2 FATs, as defined in NumberOfFats field. ActiveFat field in VolumeFlags in the Main Boot Sector determines which FAT is active.

The first cluster is cluster 2, as in FAT32. Each FatEntry represents one cluster

In exFAT, FAT is not used for tracking an allocation; an Allocation Bitmap is used for this purpose. FAT is only used for keeping chains of clusters of fragmented files. If a file is not fragmented, FAT table does not need to be updated. A Stream Extensions Directory Entry should be consulted to determine if the FAT chain is valid or not. If FAT chain is not valid, it does not need to be zeroed.

Offset	Size	Description	Comments
0 (0x00)	4	FatEntry[0]	Media type (should be 0xFFFFFFF8)
4 (0x04)	4	FatEntry[1]	Must be 0xFFFFFFFF
8 (0x08)	4	FatEntry[2]	First cluster
…	…	…	…
(ClusterCount + 1) * 4	4	FatEntry[ClusterCount + 1]	Last cluster
(ClusterCount + 2) * 4	Remainder of sector	ExcessSpace	

Valid values of FAT entries:

0x00000002
: ClusterCount +1 (max 0xFFFFFFF6) – next cluster in the chain

0xFFFFFFF7
: bad cluster

0xFFFFFFF8
: media descriptor

0xFFFFFFFF
: end of file (EOF mark)

Value 0x00000000 does not mean the cluster is free, it is an undefined value.

The second FAT table (presents only in TexFAT) is located immediately after the first one and has the same size.

Related concepts

Extended File System (exFAT) on page 167
Understanding of underlying mechanisms of data storage, organization and data recovery.

exFAT Directory Structure on page 173
Understanding of underlying mechanisms of data storage, organization and data recovery.

exFAT Defined Directory Entries on page 175
Understanding of underlying mechanisms of data storage, organization and data recovery.

exFAT Cluster Heap on page 182
Understanding of underlying mechanisms of data storage, organization and data recovery.

exFAT Directory Structure

Understanding of underlying mechanisms of data storage, organization and data recovery.

exFAT uses tree structure to describe relationship between files and directories. The root of the directory tree is defined by directory located at RootDirectoryCluster. Subdirectories are single-linked to there parents. There is no special (.) and (..) directories pointing to itself and to parent like in FAT16/FAT32.

Each directory consists of a series of directory entries. Directory entries are classified as critical/benign and primary/secondary as follows:

- Primary Directory Entries
- Critical Primary Entries
- Benign Primary Entries
- Secondary Directory Entries
- Critical Secondary Entries
- Benign Secondary Entries

Critical entries are required while benign entries are optional. Primary directory entries correspond to the entries in file system and describe main characteristics. Secondary directory entries extend the metadata associated with a primary directory entry end follow it. A group of primary/secondary entries make up a directory entry set describing a file or directory. The first directory entry in the set is a primary directory entry. All subsequent entries, if any, must be secondary directory entries.

Each directory entry derives from Generic Directory Entry template. Size of directory entry is 32 bytes.

Table 13: Generic Directory Entry Template

Offset	Size	Description	Comments
0 (0x00)	1	EntryType (see below)	
1 (0x01)	19	CustomDefined	
20 (0x14)	4	FirstCluster	0 – no cluster allocation 2..ClusterCount+1 – cluster index
24 (0x18)	8	DataLength	In bytes

Table 14: Entry Types description

Bits	Size	Description	Comments
0-4	5	Code	
5	1	Importance	0 – Critical entry, 1 – Benign entry

Bits	Size	Description	Comments
6	1	Category	0 – Primary entry, 1 – Secondary entry
7	1	In use status	0 – Not in use, 1 – In use

Entry Type can have the following values:

- **0x00** – End Of Directory marker. All other fields in directory entry are invalid. All subsequent directory entries are also End Of Directory markers
- **0x01-0x7F** (InUse = 0). All other fields in this entry are not defined
- **0x81-0xFF** (InUse = 1). Regular record with all fields defined.

Table 15: Generic Primary Directory Entry Template

Offset	Size	Description	Comments
0 (0x00)	1	EntryType	
1 (0x01)	1	SecondaryCount	Number of secondary entries which immediately follow this primary entry and together comprise a directory entry set. Valid value is 0..255
2 (0x02)	2	SetChecksum	Checksum of all directory entries in the given set excluding this field. See EntrySetCheckSum().
4 (0x04)	2	GeneralPrimaryFlags (see below)	
6 (0x06)	14	CustomDefined	
20 (0x14)	4	FirstCluster	
24 (0x18)	8	DataLength	

Bits	Size	Description	Comments
0	1	AllocationPossible	0-not possible (FirstCluster and DataLength undefined), 1-possible
1	1	NoFatChain	0-FAT cluster chain is valid 1-FAT cluster chain is not used (contiguous data)
2	14	CustomDefined	

All critical primary directory entries are located in root directory (except file directory entries). Benign primary directory entries are optional. If one benign primary entry is not recognized, all directory entry set is ignored.

```
// data points to directory entry set in memory
```

```
UINT16 EntrySetChecksum(const unsigned char data[], int secondaryCount)
{
 UINT16 checksum = 0;
 int bytes = (secondaryCount + 1) * 32;

 for (int i = 0; i < bytes; i++)
 {
  if (i == 2 || i == 3)
   continue;
  checksum = (checksum << 15) | (checksum >> 1) + data[i];
 }
 return checksum;
}
```

Related concepts

Extended File System (exFAT) on page 167
Understanding of underlying mechanisms of data storage, organization and data recovery.

Volume Layout on page 168
Understanding of underlying mechanisms of data storage, organization and data recovery.

exFAT Defined Directory Entries on page 175
Understanding of underlying mechanisms of data storage, organization and data recovery.

exFAT Cluster Heap on page 182
Understanding of underlying mechanisms of data storage, organization and data recovery.

exFAT Defined Directory Entries

Understanding of underlying mechanisms of data storage, organization and data recovery.

Main exFAT directory entries defined in table below:

Table 16: Defined Directory Entries list

Entry Type	Primary	Critical	Code	Directory Entry Name
0x81	state: available=yes	state: available=yes	1	Allocation Bitmap
0x82	state: available=yes	state: available=yes	2	Up-case Table
0x83	state: available=yes	state: available=yes	3	Volume Label
0x85	state: available=yes	state: available=yes	5	File
0xA0	state: available=yes	state: available=no	0	Volume GUID
0xA1	state: available=yes	state: available=no	1	TexFAT Padding
0xA2	state: available=yes	state: available=no	2	Windows CE Access Control Table
0xC0	state: available=no	state: available=yes	0	Stream Extension
0xC1	state: available=no	state: available=yes	1	File Name

Read about Directory entries below:

- Allocation Bitmap Directory Entry on page 176
- Up-Case Table Directory Entry on page 176
- #unique_223/unique_223_Connect_42_volum
- File Directory Entry on page 177
- Volume GUID Directory Entry on page 179
- exFAT Padding Directory Entry on page 180
- Windows CE Access Control Table Directory Entry on page 180

- #unique_223/unique_223_Connect_42_stream
- File Name Directory Entry on page 181

Allocation Bitmap Directory Entry

Offset	Size	Description	Comments
0 (0x00)	1	Entry type	0x81
1 (0x01)	1	BitmapFlags (see below)	Indicates which Allocation Bitmap the given entry describes
2 (0x02)	18	Reserved	
20 (0x14)	4	First Cluster	
24 (0x18)	8	Data Length	

Table 17: Bitmap Flags

Bits	Size	Description	Comments
0	1	BitmapIdentifier	0 – 1st bitmap, 1 - 2nd bitmap
1	7	Reserved	

The number of bitmaps and therefore a number of Bitmap Allocation entries is equal to the number of FATs. In case of TexFAT two FATs are used and bit 0 of Flags indicates which bitmap and FAT are referred.

The First Allocation Bitmap shall be used in conjunction with the First FAT and the Second Allocation Bitmap shall be used with the Second FAT. ActiveFat field in Boot Sector defines which FAT and Allocation Bitmap are active.

Bitmap size in bytes must be a number of clusters in the volume divided by 8 and rounded up.

Up-Case Table Directory Entry

Offset	Size	Description	Comments
0 (0x00)	1	Entry type	0x82
1 (0x01)	3	Reserved1	
4 (0x04)	4	TableChecksum	Up-case Table checksum
8 (0x08)	12	Reserved2	
20 (0x14)	4	FirstCluster	
24 (0x18)	8	DataLength	

The checksum is calculated against DataLength bytes of Up-case Table according to the following code:

```
UINT32 UpCaseTableChecksum(const unsigned char data[], int bytes)
{
 UINT32 checksum = 0;

 for (int i = 0; i < bytes; i++)
  checksum = (checksum << 31) | (checksum >> 1) + data[i];

 return checksum;
}
```

Volume Label Directory Entry

Offset	Size	Description	Comments
0 (0x00)	1	Entry type	0x83
1 (0x01)	1	CharacterCount	Length in Unicode characters (max 11)
2 (0x02)	22	VolumeLabel	Unicode string
24 (0x18)	8	Reserved	

If volume is formatted without a label, the Volume Label Entry will be present but Entry Type will be set to 0x03 (not in use).

File Directory Entry

File directory entry describes files and directories. It is a primary critical directory entry and must be immediately followed by 1 Stream Extension directory entry and from 1 to 17 File Name directory entries. Those 3-19 directory entries comprise a directory entry set describing a single file or a directory.

Offset	Size	Description	Comments
0 (0x00)	1	Entry type	0x85
1 (0x01)	1	SecondaryCount	Must be from 2 to 18
2 (0x02)	2	SetChecksum	
4 (0x04)	2	FileAttributes (see below)	
6 (0x06)	2	Reserved1	
8 (0x08)	4	CreateTimestamp	
12 (0x0C)	4	LastModifiedTimestamp	
16 (0x10)	4	LastAccessedTimestamp	
20 (0x14)	1	Create10msIncrement	0..199
21 (0x15)	1	LastModified10msIncrement	0..199
22 (0x16)	1	CreateTimezoneOffset	Offset from UTC in 15 min increments
23 (0x17)	1	LastModifiedTimezoneOffset	Offset from UTC in 15 min increments
24 (0x18)	1	LastAccessedTimezoneOffset	Offset from UTC in 15 min increments
25 (0x19)	7	Reserved2	

Table 18: File Attributes

Bits	Size	Description	Comments
0	1	ReadOnly	
1	1	Hidden	
2	1	System	
3	1	Reserved1	

Bits	Size	Description	Comments
4	1	Directory	
5	1	Archive	
6	10	Reserved2	

Table 19: Time stamp Format

Bits	Size	Description	Comments
0-4	5	Seconds (as number of 2-second intervals)	0..29 29 represents 58 seconds
5-10	6	Minutes	0..59
11-15	5	Hour	0..23
16-20	5	Day	1..31
21-24	4	Month	1..12
25-31	7	Year (as offset from 1980)	0 represents 1980

Time stamp format records seconds as 2 seconds intervals, so 10ms increments are used to increase precision from 2 seconds to 10 milliseconds. The valid values are from 0 to 199 in 10ms intervals which are added to correspondent time stamp. Time stamp is recorded in local time.

Time zone offset is expressed in 15 minutes increments.

Table 20: Time Zone Offset Tablet

Timezone Offset field	TZ Offset	Time Zone	Comments
128 (0x80)	UTC	Greenwich Standard Time	
132 (0x84)	UTC+01:00	Central Europe Time	
136 (0x88)	UTC+02:00	Eastern Europe Standard Time	
140 (0x8C)	UTC+03:00	Moscow Standard Time	
144 (0x90)	UTC+04:00	Arabian Standard Time	
148 (0x94)	UTC+05:00	West Asia Standard Time	
152 (0x98)	UTC+06:00	Central Asia Standard Time	
156 (0x9C)	UTC+07:00	North Asia Standard Time	
160 (0xA0)	UTC+08:00	North Asia East Standard Time	
164 (0xA4)	UTC+09:00	Tokyo Standard Time	
168 (0xA8)	UTC+10:00	West Pacific Standard Time	
172 (0xAC)	UTC+11:00	Central Pacific Standard Time	

Timezone Offset field	TZ Offset	Time Zone	Comments
176 (0xB0)	UTC+12:00	New Zealand Standard Time	
180 (0xB4)	UTC+13:00	Tonga Standard Time	
208 (0xD0)	UTC-12:00	Dateline Standard Time	
212 (0xD4)	UTC-11:00	Samoa Standard Time	
216 (0xD8)	UTC-10:00	Hawaii Standard Time	
220 (0xDC)	UTC-09:00	Alaska Standard Time	
224 (0xE0)	UTC-08:00	Pacific Standard Time	
228 (0xE4)	UTC-07:00	Mountain Standard Time	
232 (0xE8)	UTC-06:00	Central Standard Time	
236 (0xEC)	UTC-05:00	Eastern Standard Time	
240 (0xF0)	UTC-04:00	Atlantic Standard time	
242 (0xF2)	UTC-03:30	Newfoundland Standard Time	
244 (0xF4)	UTC-03:00	Greenland Standard Time	
248 (0xF8)	UTC-02:00	Mid-Atlantic Standard Time	
252 (0xFC)	UTC-01:00	Azores Standard Time	

Volume GUID Directory Entry

In following table presented a benign primary directory entry and may not present in a file system.

Offset	Size	Description	Comments
0 (0x00)	1	EntryType	0xA0
1 (0x01)	1	SecondaryCount	Must be 0x00
2 (0x02)	2	SetChecksum	
4 (0x04)	2	GeneralPrimaryFlags (See below)	
6 (0x06)	16	VolumeGuid	All values are valid except null GUID {00000000-0000-0000-0000-00000000
22 (0x16)	10	Reserved	

Table 21: Primary Flags Definitions

Bits	Size	Description	Comments
0	1	AllocationPossible	Must be 0
1	1	NoFatChain	Must be 0
2	14	CustomDefined	

exFAT Padding Directory Entry

Offset	Size	Description	Comments
0 (0x00)	1	EntryType	0xA1
1 (0x01)	31	Reserved	

Remember:

exFAT 1.00 does not define TexFAT Padding directory entry. TexFAT Padding directory entries are only valid in the first cluster of directory and occupy every directory entry of the cluster. The implementations should not move TexFAT Padding directory entries.

Windows CE Access Control Table Directory Entry

Offset	Size	Description	Comments
0 (0x00)	1	EntryType	0xA2
1 (0x01)	31	Reserved	

Remember:

exFAT 1.00 does not define Windows CE Access Control Table Directory Entry.

Stream Extension Directory Entry

Offset	Size	Description	Comments
0 (0x00)	1	EntryType	0xC0
1 (0x01)	1	GeneralSecondaryFlags (see below)	
2 (0x02)	1	Reserved1	
3 (0x03)	1	NameLength	Length of Unicode name contained in subsequent File Name directory entries
4 (0x04)	2	NameHash	Hash of up-cased file name
6 (0x06)	2	Reserved2	
8 (0x08)	8	ValidDataLength	Must be between 0 and DataLength
16 (0x10)	4	Reserved3	
20 (0x14)	4	FirstCluster	
24 (0x18)	8	DataLength	For directories maximum 256 MB

Table 22: Secondary Flags Definitions

Bits	Size	Description	Comments
0	1	AllocationPossible	Must be 1
1	1	NoFatChain	
2	14	CustomDefined	

Stream Extension directory entry must immediately follow the File directory entry in the set. It could be only one Stream Extension entry in the set. If NoFatChain flag is set, all allocated clusters are contiguous.

The NameHash field facilitates the purpose of fast file name comparison and is performed on up-cased file name. NameHash verify against a mismatch, however matching hashes cannot guarantee the equality of file names. If name hashes match, a subsequent full name comparison must be performed.

```
// fileName points to up-cased file name
UINT16 NameHash(WCHAR *fileName, int nameLength)
{
  UINT16 hash = 0;
  unsigned char *data = (unsigned char *)fileName;

  for (int i = 0; i < nameLength * 2; i++)
    hash = (hash << 15) | (hash >> 1) + data[i];

  return hash;
}
```

ValidDataLength determines how much actual data written to the file. Implementation shall update this field as data has been written. The data beyond the valid data length is undefined and implementation shall return zeros.

File Name Directory Entry

Offset	Size	Description	Comments
0 (0x00)	1	EntryType	0xC1
1 (0x01)	1	GeneralSecondaryFlags (see below)	
2 (0x02)	30	FileName	

Table 23: Secondary Flags Definitions

Bits	Size	Description	Comments
0	1	AllocationPossible	Must be 0
1	1	NoFatChain	Must be 0
2	14	CustomDefined	

File Name directory entries must immediately follow the Steam Extension directory entry in the number of NameLength/15 rounded up. The maximum number of File Name entries is 17, each can hold up to 15 Unicode characters and the maximum file name length is 255. Unused portion of FileName field must be set to 0x0000.

Table 24: Invalid File Name Characters

Character Code	Character	Description
0x0000 – 0x001F		Control codes
0x0022	"	Quotation mark
0x002A	*	Asterisk
0x002F	/	Forward slash
0x003A	:	Colon
0x003C	<	Less than
0x003E	>	Greater than
0x003F	?	Question mark
0x005C	\	Back slash
0x007C	|	Vertical bar

Related concepts

Extended File System (exFAT) on page 167
Understanding of underlying mechanisms of data storage, organization and data recovery.

Volume Layout on page 168
Understanding of underlying mechanisms of data storage, organization and data recovery.

exFAT Directory Structure on page 173
Understanding of underlying mechanisms of data storage, organization and data recovery.

exFAT Cluster Heap on page 182
Understanding of underlying mechanisms of data storage, organization and data recovery.

exFAT Cluster Heap

Understanding of underlying mechanisms of data storage, organization and data recovery.

The cluster heap is a set of clusters which hold data in exFAT. It contains:

- Root Directory
- Files
- Directories
- Allocation Bitmap on page 182
- Up-case Table on page 183

The allocation status of clusters in cluster heap is tracked by Bitmap Allocation Table which itself located inside the cluster heap.

Allocation Bitmap

Allocation Bitmap keeps track of the allocation status of clusters. FAT does not serve this purpose as in FAT16/FAT32 file system. Allocation Bitmap consists of a number of 8 bit bytes which can be treated as a sequence of bits. Each bit in bitmap corresponds to a data cluster. If it has a value of 1, the cluster is occupied, if 0 - the cluster is free. The least significant bit of bitmap table refers to the first cluster, i.e. cluster 2.

Offset	Size	Description	Comments
0x00	1	1st byte	Clusters 2-9

Offset	Size	Description	Comments
0x01	1	2nd byte	Clusters 10-17
0x02	1	3rd byte	Clusters 18-25
...			

Bitmap allocation table resides in cluster heap and referred by Bitmap Directory entry in root directory.

In exFAT could be 2 Bitmap Allocation tables, otherwise there will be only one bitmap. The NumberOfFats field in Boot Sectors determines the number of valid Allocation Bitmap directory entries in the root directory and the number of Allocation Bitmaps.

Up-case Table

Up-case table contains data used for conversion from lower-case to upper-case characters. File Name Directory Entry uses Unicode characters and preserves case when storing file name. exFAT itself is case insensitive, so it needs to compare file names converted to the upper-case during search operations.

Normally Up-case table is located right after Bitmap Allocation table but can be placed anywhere is the cluster heap. It has a corresponding primary critical directory entry in the root directory.

Up-case Table is an array of Unicode characters, an index of which represents the Unicode characters to be up-cased and the value is the target up-cased character. The Up-case Table shall contain at least 128 mandatory Unicode mappings. If implementation supports only mandatory 128 characters it may ignore the rest of Up-case Table. When up-casing file names such implementation shall up-case only characters from the mandatory 128 characters set and leave other characters intact. When comparing file names which are different only by characters in non-mandatory set, those file names shall be treated as equal.

Index	Value	Comments
0x0000	0x0000	
0x0001	0x0001	
0x0002	0x0002	
...	...	..
0x0041	0x0041	'A' is mapped into itself (identity mapping)
0x0042	0x0042	'B' is mapped into itself
..	..	..
0x061	0x041	'a' is mapped into 'A' (non-identity mapping)
0x062	0x0042	'b' is mapped into 'B'
..	..	..

Up-case Table can be written in compressed format where the series of identity mappings is represented with 0xFFFF followed by the number of identity mappings.

```
Mandatory First 128 Up-case Table Entries

Index |  Table Entries
_____________________________________________________________________________
0000  -  0000 0001 0002 0003 0004 0005 0006 0007 0008 0009 000A 000B 000C 000D 000E 000F
0010  -  0010 0011 0012 0013 0014 0015 0016 0017 0018 0019 001A 001B 001C 001D 001E 001F
0020  -  0020 0021 0022 0023 0024 0025 0026 0027 0028 0029 002A 002B 002C 002D 002E 002F
0030  -  0030 0031 0032 0033 0034 0035 0036 0037 0038 0039 003A 003B 003C 003D 003E 003F
0040  -  0040 0041 0042 0043 0044 0045 0046 0047 0048 0049 004A 004B 004C 004D 004E 004F
0050  -  0050 0051 0052 0053 0054 0055 0056 0057 0058 0059 005A 005B 005C 005D 005E 005F
0060  -  0060 0041 0042 0043 0044 0045 0046 0047 0048 0049 004A 004B 004C 004D 004E 004F
```

```
0070  -  0050 0051 0052 0053 0054 0055 0056 0057 0058 0059 005A 007B 007C 007D 007E 007F
```

Remember:

Non-identity mappings are highlighted in **bold**.

```
Mandatory First 128 Up-case Table Entries in compressed format

Index |  Table Entries
_____________________________________________________________________________
0000  -  FFFF 0061 0041 0042 0043 0044 0045 0046 0047 0048 0049 004A 004B 004C 004D 004E
0010  -  004F 0050 0051 0052 0053 0054 0055 0056 0057 0058 0059 005A FFFF 0005
```

The first highlighted group describes that first 0x0061 characters (0x0000-0x0060) have identity mappings. The next character after it (0x0061) maps to 0x0041 etc. until the next compressed group is encountered.

Remember:

The first highlighted **in bold** group describes that first 0x0061 characters (0x0000-0x0060) have identity mappings. The next character after it (0x0061) maps to 0x0041 etc. until the next compressed group is encountered.

Related concepts

Extended File System (exFAT) on page 167
Understanding of underlying mechanisms of data storage, organization and data recovery.

Volume Layout on page 168
Understanding of underlying mechanisms of data storage, organization and data recovery.

exFAT Directory Structure on page 173
Understanding of underlying mechanisms of data storage, organization and data recovery.

exFAT Defined Directory Entries on page 175
Understanding of underlying mechanisms of data storage, organization and data recovery.

Erase disk concept

Understanding of underlying mechanisms of data storage organization and data erasure.

Modern methods of data encryption are deterring network attackers from extracting sensitive data from stored database files.

Attackers (who want to retrieve confidential data) become more resourceful and look for places where data might be stored temporarily. For example, the Windows ***DELETE*** command merely changes the files attributes and location so that the operating system will not look for the file located on FAT/exFAT volumes. The situation with NTFS file system is similar.

One avenue of attack is the recovery of data from residual data on a discarded hard drive. When deleting confidential data from hard drives, removable disks or USB devices, it is important to extract all traces of the data so that recovery is not possible.

Most official guidelines regarding the disposal of confidential magnetic data do not take into account the depth of today's recording densities nor the methods used by the OS when removing data.

Removal of confidential personal information or company trade secrets in the past might have been performed using the ***FORMAT*** command or the ***FDISK*** command. Using these procedures gives users a sense of confidence that the data has been completely removed.

When using the ***FORMAT*** command Windows displays a message like this: `Formatting a disk removes all information from the disk.`

Actually the ***FORMAT*** utility creates new empty directories at the root area, leaving all previous data on the disk untouched. Moreover, an image of the replaced FAT tables is stored so that the ***UNFORMAT*** command can be used to restore them.

FDISK merely cleans the Partition Table (located in the drive's first sector) and does not touch anything else.

Moreover, most of hard disks contain hidden zones (disk areas that cannot be accessed and addressed on a logical access level).

Active@ KillDisk conforms to more than 20 international standards for clearing and sanitizing data (US DoD 5220.22-M, Gutmann and others). You can be sure that sensitive information is destroyed forever once you erase a disk with KillDisk.

Active@ KillDisk is a professional security application that destroys data permanently on any computer that can be started using a bootable CD/DVD/BD or USB Flash Disk. Access to the drive's data is made on the physical level via the BIOS (Basic Input-Output System) bypassing the operating system's logical drive structure organization. Regardless of the operating system, file systems, or type of machine, this utility can destroy all the data on all storage devices. It does not matter which operating systems or file systems are located on the machine.

Related information

Sanitization Types on page 6
Disk Hidden Zones on page 202

Secure Erase concepts

Secure Erase for SSD is used to permanently delete data from the media and to restore the drive's speed if it starts to drop to noticeably lower performance than stated (at the same time, we don't consider SLC-caching and other "official" reasons for speed reduction since it's hardware drive features).

The essence of the problem that Secure Erase can solve: drive began to work slowly (writing and reading data). There can be a lot of reasons, some of them are related to the hardware component and some to the software component. SSDs are very different in service from classic HDDs, therefore, simply deleting data or formatting the drive does not really mean resetting the cell - you need to clear it before recording, which slows down the process of recording new data. In theory, there shouldn't be such problems, because TRIM exists - a command to clear the data marked for deletion in cells. This command only works with 2.5" and M.2 SATA drives. For drives connected to the PCIe bus (M.2 or PCIe on the motherboard) there is an analogue - Deallocate. But it happens that these functions are disabled for some reason - an OS error, a user error in setting up a disk through third-party software, or the use of non-standard OS assemblies with unknown software components. So, the disk starts to work noticeably slower and it is quite noticeable without any benchmark performance measurements.

SSDs use a number of mapping layers that hide the physical layout of the flash-based memory, as well as help in managing how flash memory data integrity and lifetime are managed. Collectively, these layers are referred to as the Flash Translation Layer (FTL).

SSDs are also over-provisioned: they contain a bit more flash memory than what they're rated for. This extra memory is used internally by the FTL as empty data blocks, used when data needs to be rewritten, and as out-of-band sections for use in the logical to physical mapping.

The mapping layers, and how the flash controller manages memory allocation, pretty much ensure that either erasing or performing a conventional hard drive type of secure erase won't ensure all data is overwritten, or even erased at all.

One example of how data gets left behind intact is due to how data is managed in an SSD. When you edit a document and save the changes, the saved changes don't overwrite the original data (an in-place update). Instead, SSDs write the new content to an empty data block and then update the logical to physical map to point to the new location. This leaves the space the original data occupied on the SSD marked as free,

but the actual data is left intact. In time, the data marked as free will be reclaimed by the SSD's garbage collection system, but until then, the data could be recovered.

A conventional Secure Erase, as used with hard drives, is unable to access all of the SSD's memory location, due to the FTL and how an SSD actually writes data, which could lead to intact data being left behind.

SSD manufacturers understand the need for an easy way to sanitize an SSD, and most have implemented the ATA command, Secure Erase Unit (used with SATA-based SSDs), or the NVMe command, Format NVM (used with PCIe-based SSDs) as a fast and effective method of securely erasing an SSD.

So, SSD drives have a non-trivial system of work, therefore, the scheme for the complete destruction of data should also not be the easiest. But in reality, this is not so at all. Any SSD has a controller that is the "brain" of the drive. He not only tells the system where to write data, but also encrypts the information passing through it and stores the key with himself. If you remove (or rather replace) a given key, then all the information will turn into a random set of 1 and 0 - it will be impossible to decrypt it in any way. Just one simple action by the user can solve the problem of safe data erasure. This method is the fastest and most effective.

Note:

To protect information that is critical, both for serious organizations that are concerned about the safety of data and for public sector enterprises working with information classified as state secrets, information systems should usually use certified sanitation algorithms (*US DoD 5220.22-M*, *Canadian OPS-II*, *NSA 130-2* etc.).

If you combine these two methods (replacing the key and resetting the cells), you get the perfect algorithm for obtaining a completely sterile disk in the state of its maximum performance. This, firstly, solves the problem that we raised at the very beginning, and, secondly, it can help us answer the question about the degree of drive wear.

It is important to note that some drives with built-in encryption can receive only one algorithm upon receipt of a safe erase command - it depends on the controller settings by the manufacturer. If you "reset" your SSD and compare the actual performance with the declared one, you will get the answer to this question. This procedure does not affect disk wear (which is very important). Note that these actions are designed specifically for analyzing the state of the disk, but it will not be possible to achieve a long-term increase in the read/write speed due to the peculiarities of the operation of SSD disks - the situation may depend on both the drive model and the controller firmware. And it must be noted that not all drives support encryption. In this case, the controller simply resets the cells.

Erase Methods

One Pass Zeros or One Pass Random

When using One Pass Zeros or One Pass Random standard, the number of passes is fixed and cannot be changed. When the write head passes through a sector, it writes only zeros or a series of random characters.

US DoD 5220.22-M

The write head passes over each sector three times. The first time with zeros 0x00, second time with 0xFF and the third time with random characters. There is one final pass to verify random characters by reading.

US DoE M205.1-2

The write head passes over each sector three times. The first time with random characters, second time with other random characters and the third time with zeros 0x00. There is one final pass to verify zeroes 0x00 by reading.

Canadian CSEC ITSG-06

The write head passes over each sector, writing a random character. On the next pass, writes the compliment of previously written character. Final pass is random, proceeded by a verify.

Canadian OPS-II
The write head passes over each sector seven times (0x00, 0xFF, 0x00, 0xFF, 0x00, 0xFF, random). There is one final pass to verify random characters by reading.

British HMG IS5 Baseline
Baseline method overwrites disk's surface with just zeros 0x00. There is one final pass to verify random characters by reading.

British HMG IS5 Enhanced
Enhanced method - the write head passes over each sector three times. The first time with zeros 0x00, second time with 0xFF and the third time with random characters. There is one final pass to verify random characters by reading.

Russian GOST p50739-95
The write head passes over each sector two times: 0x00, Random. There is one final pass to verify random characters by reading.

US Army AR380-19
The write head passes over each sector three times. The first time with 0xFF, second time with zeros 0x00 and the third time with random characters. There is one final pass to verify random characters by reading.

US Air Force 5020
The write head passes over each sector three times. The first time with random characters, second time with zeros 0x00 and the third time with 0xFF. There is one final pass to verify random characters by reading.

NAVSO P-5329-26 RL
RL method - the write head passes over each sector three times: 0x01, 0x27FFFFFF, Random. There is one final pass to verify random characters by reading.

NCSC-TG-025
The write head passes over each sector three times: 0x00, 0xFF, Random. There is one final pass to verify random characters by reading.

NSA 130-2
The write head passes over each sector two times: Random, Random. There is one final pass to verify random characters by reading.

NIST 800-88
Supported three NIST 800-88 media sanitation standards:

- 1. The write head passes over each sector one time (0x00).
- 2. The write head passes over each sector one time (Random).
- 3. The write head passes over each sector three times (0x00, 0xFF, Random).

For details about this,the most secure data clearing standard, you can read the original article at the link below: http://csrc.nist.gov/publications/nistpubs/800-88/NISTSP800-88_with-errata.pdf

German VSITR
The write head passes over each sector seven times.

Bruce Schneier
The write head passes over each sector seven times: 0xFF, 0x00, Random, Random, Random, Random, Random. There is one final pass to verify random characters by reading.

Peter Gutmann
The write head passes over each sector 35 times. For details about this, the most secure data clearing standard, you can read the original article: http://www.cs.auckland.ac.nz/%7Epgut001/pubs/se%0Acure_del.html

Australian ISM-6.2.93
The write head passes over each sector once with random characters. There is one final pass to verify random characters by reading.

IEEE Std 2883-2022
IEEE Std 2883-2022 clear sanitization method consists of at least two passes of writes, to include a pattern in the first pass and its complement in the second pass. (0x00 in the first pass and 0xFF in the second). Verification step selects random locations on the storage media that represent at least 5% of the addressable space.

Secure Erase (ANSI ATA, SE)
According to National Institute of Standards and Technology (NIST) Special Publication 800-88: Guidelines for Media Sanitation, *Secure Erase* is "*An overwrite technology using firmware based process to overwrite a hard drive. Is a drive command defined in the ANSI ATA and SCSI disk drive interface specifications, which runs inside drive hardware. It completes in about 1/8 the time of 5220 block erasure.*" The guidelines also state that "*degaussing and executing the firmware Secure Erase command (for ATA drives only) are acceptable methods for purging.*" ATA Secure Erase (SE) is designed for SSD controllers. The SSD controller resets all memory cells making them empty. In fact, this method restores the SSD to the factory state, not only deleting data but also returning the original performance. When implemented correctly, this standard processes all memory, including service areas and protected sectors.

User Defined
User indicates the number of times the write head passes over each sector. Each overwriting pass is performed with a buffer containing user-defined or random characters. User Defined method allows to define any kind of new erase algorithms based on user requirements.

US DoD 5220.22-M erase method
The **U.S. Department of Defense (DoD) 5220.22-M** erase method is a **data sanitization standard** historically defined in the **National Industrial Security Program Operating Manual (NISPOM)**. It specifies a process for **securely overwriting data on magnetic storage media** (e.g., hard drives, tapes) to prevent recovery of sensitive information.

Although it's no longer the official DoD standard (and has been superseded by newer guidelines such as **NIST SP 800-88 Rev.1**), it remains one of the most referenced and implemented data-erasure methods in commercial software.

Technical Description

The DoD 5220.22-M method involves **multiple overwrite passes** using specific bit patterns:

Common 3-pass implementation:

Pass 1: Overwrite all addressable locations with **binary zeros (0x00)**;
Pass 2: Overwrite all addressable locations with **binary ones (0xFF)**;

Pass 3: Overwrite all addressable locations with **a random character (pseudo-random data)**, and optionally verify the write;

Variations:

- Some versions (e.g., **DoD 5220.22-M (ECE)**) add **a verify step after each pass** to confirm successful writing.
- Others use **seven passes**, alternating between 0s, 1s, and random data to further reduce the chance of magnetic residue being recoverable.

Underlying Principle

The goal is to **overwrite magnetic domains** on a storage medium enough times that the original bit patterns (residual magnetization) are effectively destroyed. On older magnetic drives, forensic recovery of overwritten data was theoretically possible, which motivated multi-pass overwriting.

Modern drives, especially SSDs, use wear-leveling and remapping — meaning overwriting may not reliably reach all physical locations. Hence, software-based erasure methods like DoD 5220.22-M are not guaranteed effective for **solid-state media**.

Use Cases

- **Government and defense organizations** (historically): To sanitize classified or sensitive data before declassification or repurposing of storage devices.
- **Corporate IT departments:** For secure data disposal, ensuring that retired or redeployed hard drives don't leak confidential data.
- **Data destruction services and erasure software:** Many commercial tools (e.g., Blancco, DBAN) implement "DoD 5220.22-M wipe" as a configurable option.
- **Legacy systems:** When physical destruction is not viable and the medium is magnetic (HDDs, tapes).

Modern Considerations

- The DoD 5220.22-M method is **deprecated** and **not recognized as a current federal standard**.
- For **current best practices**, the **NIST SP 800-88 Rev.1 "Clear, Purge, Destroy"** framework is preferred.
- For **SSDs**, cryptographic erase or manufacturer-specific secure erase commands are more effective than overwriting.

Canadian CSEC ITSG-06 erase method

The **Canadian CSEC ITSG-06** (Communications Security Establishment Canada – *Information Technology Security Guidance No. 6*) erase method is a **data sanitization guideline** issued by the **CSEC**, which governs how to securely erase information from electronic storage devices to prevent unauthorized data recovery.

It is the **Canadian government's standard** for clearing, purging, and destroying data on magnetic and solid-state media, analogous to the U.S. DoD 5220.22-M or NIST SP 800-88 standards.

Overview

Full name: *ITSG-06 — Clearing and Declassifying Electronic Data Storage Devices*

Issued by: Communications Security Establishment Canada (CSEC)

Purpose: To ensure that sensitive or classified data is irrecoverable from storage media before reuse, downgrade, or disposal.

Technical Description

The **ITSG-06 erase method** provides detailed procedures for **clearing** and **purging** magnetic media (e.g., hard drives, tapes) and **solid-state storage** (e.g., SSDs, flash drives).

While implementations can vary slightly based on media type and classification level, the general **overwrite method** for magnetic media is as follows:

Magnetic Storage (HDDs, tapes):

First pass: Overwrite all addressable storage locations with a **fixed pattern** (e.g., binary 0s).
Verification: Verify that the last overwrite pass was successful (bit-level verification on a sample or full-drive basis).
Third pass: Overwrite all addressable storage locations with **random data**.
Second pass: Overwrite all addressable storage locations with the **complementary pattern** (e.g., binary 1s).

This is effectively similar to the DoD 5220.22-M (3-pass) method, with CSEC validation requirements.

Solid-State Storage (SSDs, Flash, Hybrid Drives):

- ITSG-06 **discourages overwriting** due to wear-leveling and remapping mechanisms in SSDs.

- Instead, it recommends:
  - Using **built-in ATA Secure Erase** or **cryptographic erase** functions.
  - **Physical destruction** (e.g., shredding, degaussing, or incineration) if the device handled highly classified data.

Rationale and Effectiveness

The ITSG-06 standard is based on the principle that **residual data (remanence)** on magnetic domains or flash cells can potentially be recovered with advanced forensic techniques.

Multiple overwriting passes or hardware-supported secure erase ensures that no readable magnetic signature of the original data remains.

Use Cases

- **Canadian federal departments and agencies:**

  To securely sanitize storage media containing **Protected** or **Classified** information before re-use or release.
- **Defense contractors and critical infrastructure operators:**

  When handling Canadian government or NATO-classified data that falls under CSEC oversight.
- **Private sector and IT asset disposal firms:**

  Implementing ITSG-06 erasure processes to comply with Canadian privacy laws and data protection standards (e.g., PIPEDA).
- **Cross-border compliance:**

  Companies operating in both the U.S. and Canada may adopt ITSG-06 alongside DoD 5220.22-M or NIST 800-88 for harmonized data sanitization.

Modern Considerations

- Like DoD 5220.22-M, **ITSG-06 is being phased out** in favor of newer approaches aligned with **NIST SP 800-88 Rev.1** and **CSE's updated guidance** for data destruction.
- For **SSDs and encrypted media**, CSEC recommends **cryptographic erase** or **physical destruction** as the most reliable methods.

Summary

Media Type	Method	Passes	Verification	Notes
HDD / Tape	Overwrite fixed, complement, random	3	Optional / recommended	Standard ITSG-06 wipe
SSD / Flash	Secure Erase or Cryptographic Erase	N/A	Automatic	Overwriting not reliable
Classified Media	Physical Destruction	N/A	N/A	Required for Top Secret or higher

Wipe Disk concept

Wiping unoccupied disk's space

You may have confidential data on your hard drive in spaces where data may have been stored temporarily.

You may also have deleted files by using the Windows Recycle Bin and then emptying it. While you are still using your local hard drive, there may be confidential information available in these unoccupied spaces.

Wiping the logical drive's deleted data does not delete existing files and folders. It processes all unoccupied drive space so that recovery of previously deleted files becomes impossible.

Installed applications and existing data are not touched by this process. When you wipe unoccupied drive space, the process is run from the bootable CD/DVD operating system. As a result, the wipe or erase process uses an operating system that is outside the local hard drive and is not impeded by Windows system caching. This means that deleted Windows system records can be wiped clean.

Active@ KillDisk wipes unused data residue from file slack space, unused sectors, and unused space in MTF records or directory records.

Wiping drive space can take a long time, so do this when the system is not being otherwise utilized. For example, this can be done overnight.

Wipe algorithms

The process of deleting files does not eliminate them from the hard drive. Unwanted information may still be left available for recovery on the computer. A majority of software that advertises itself as performing reliable deletions simply wipes out free clusters. Deleted information may be kept in additional areas of a drive. **Active@ KillDisk** therefore offers different wipe algorithms to ensure secure deletion: overwriting with zeros, overwriting with random values, overwriting with multiple passes using different patterns and much more. **Active@ KillDisk** supports more than 20 international data sanitizing standards, including US DoD 5220.22M and the most secure Gutmann's method overwriting with 35 passes.

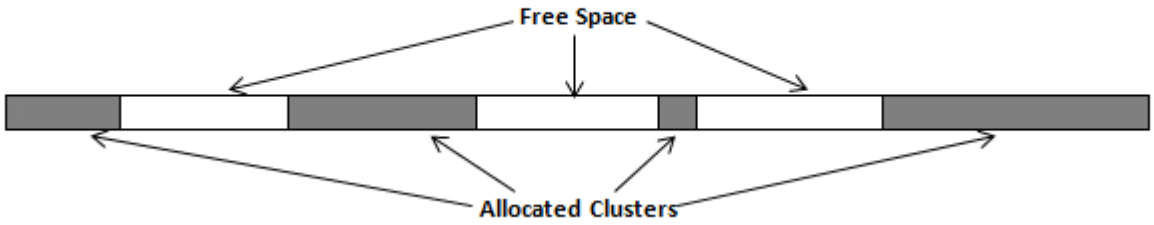


Figure 68: Disk Free Space and Allocated Clusters

Wiping file slack space

This relates to any regular files located on any file system. Free space to be wiped is found in the "tail" end of a file because disk space is usually allocated in 4 Kb clusters. Most files have sizes that are not 4 Kb increments and thus have slack space at their end.

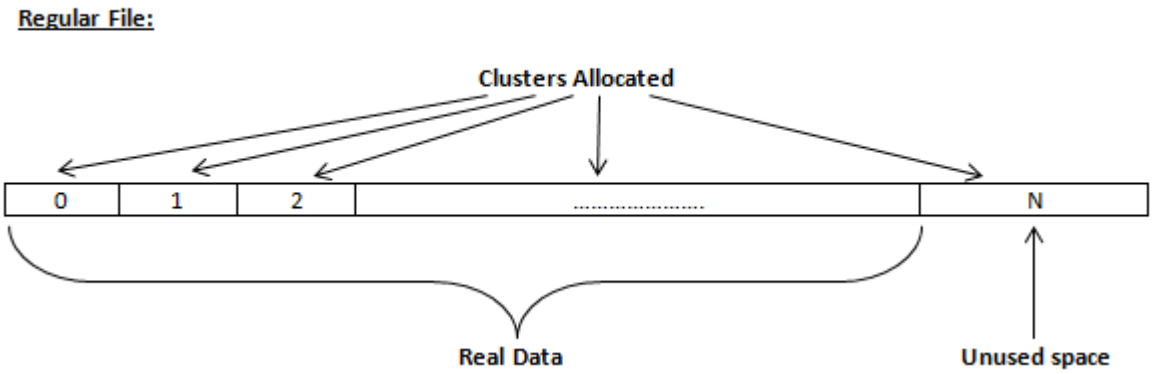


Figure 69: File Slack Space and Allocated Clusters

Specifics of wiping Microsoft NTFS file system

NTFS Compressed Files

Wiping free space inside a file: The algorithm NTFS uses to "compress" a file operates by separating the file into compressed blocks (usually 64 Kb long). After it is processed, each of these blocks has been allocated a certain amount of space on the volume. If the compressed information takes up less space than the source file, then the rest of the space is labeled as sparse space and no space on the volume is allocated to it. Because the compressed data often doesn't have a size exactly that of the cluster, the end of each of these blocks stays as unusable space of significant size. Our algorithm goes through each of these blocks in a compressed file and wipes the unusable space, erasing previously deleted information that was kept in those areas.

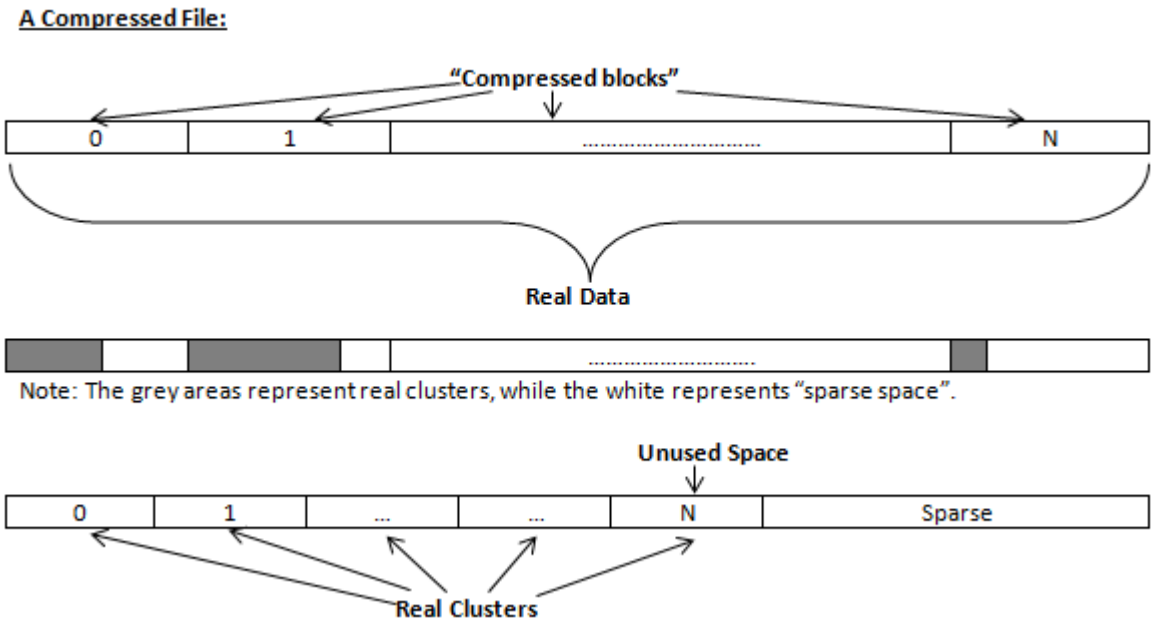


Figure 70: Compressed file structure

The MFT (Master File Table) Area

Wiping the system information:

The MFT file contains records, describing every file on the volume. During the deletion of these files, the records of their deletion are left untouched - they are simply recorded as "deleted". Therefore file recovery software can use this information to recover anything from the name of the file and the structure of the deleted directories down to files smaller than 1Kb that are able to be saved in the MFT directly. The algorithm used by **Active@ KillDisk** wipes all of the unused information out of the MFT records and wipes the unusable space, making a recovery process impossible.

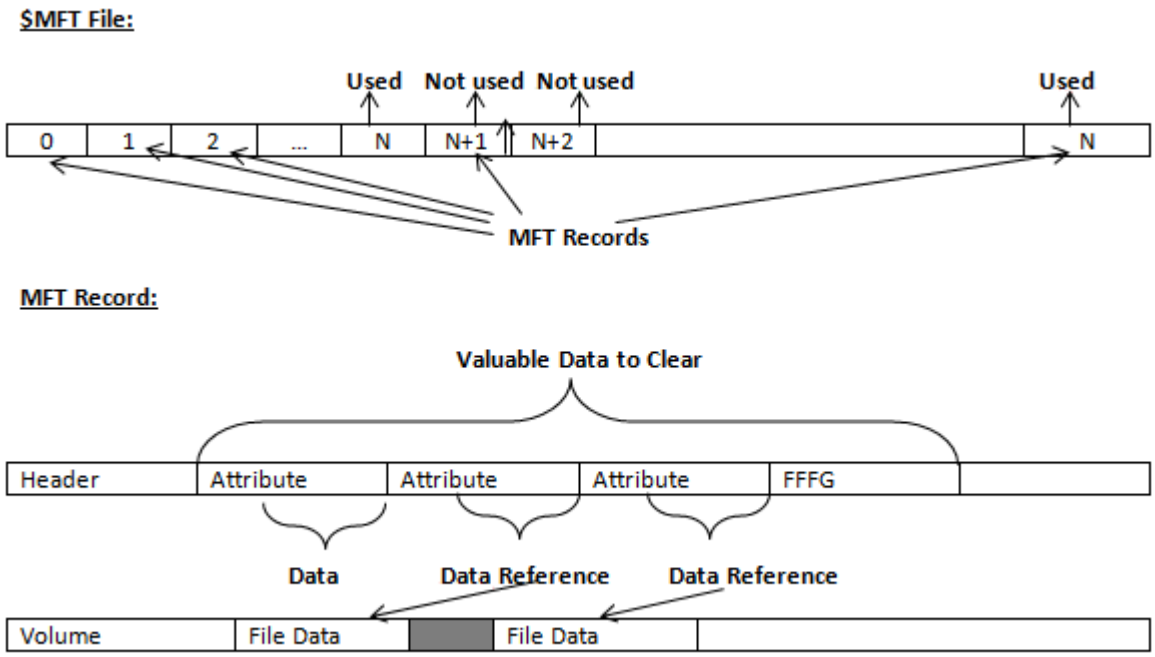


Figure 71: MFT Structure

Specifics of wiping Microsoft FAT file system

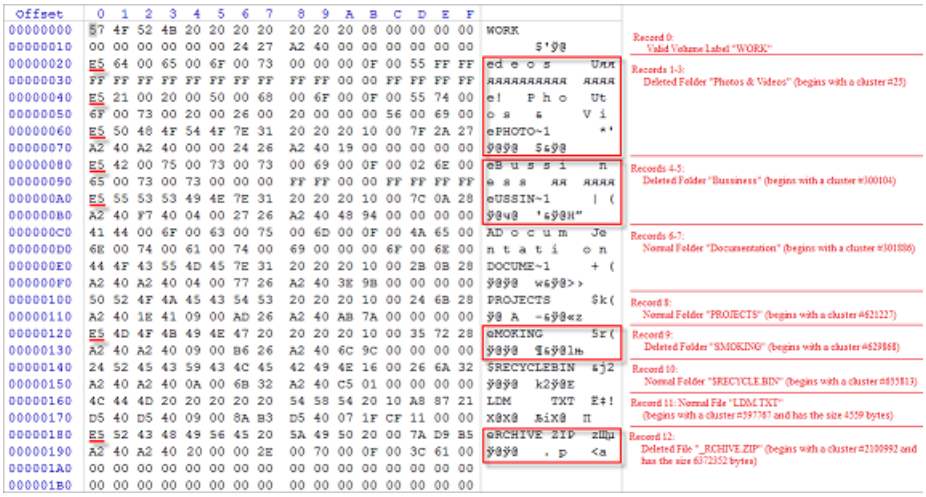
Wiping Directory Areas

Each directory on a FAT/FAT32 or an exFAT volume can be considered as a specific file, describing the contents of the directory. Inside this descriptor there are many 32-byte records, describing every file and other inner folders.

When you delete files this data is not being fully erased. It is just marked as deleted (hex symbol 0xE5). That's why data recovery software can detect and use these records to restore file names and full directory structures.

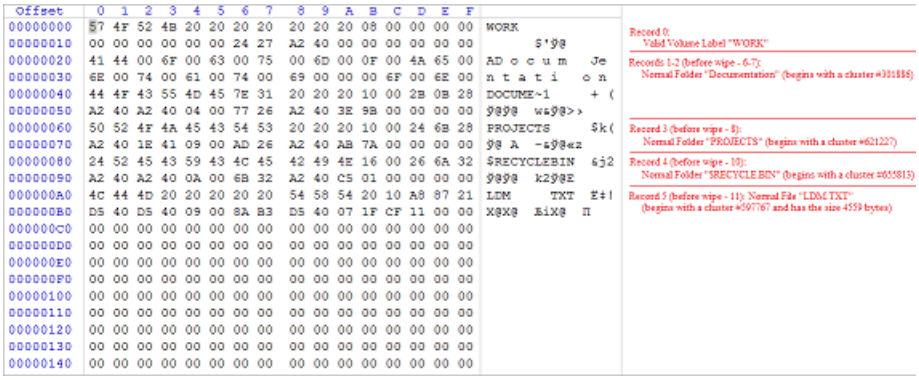
In some cases dependent on whether a space where item located has been overwritten yet or not, files and folders can be fully or partially recovered..

Active@ KillDisk makes data recovery impossible by using an algorithm that wipes out all unused information from directory descriptors. **Active@ KillDisk** not only removes unused information, but also defragments Directory Areas, thus speeding up directory access.



In this example red rectangles display deleted records.

Figure 72: FAT Directory before Wipe



In this example all deleted records removed and root folder defragmented.

Figure 73: FAT Directory after Wipe

Specifics of wiping Apple HFS+ file system

HFS+ B-tree

A B-tree file is divided up into fixed-size nodes, each of which contains records consisting of a key and some data.

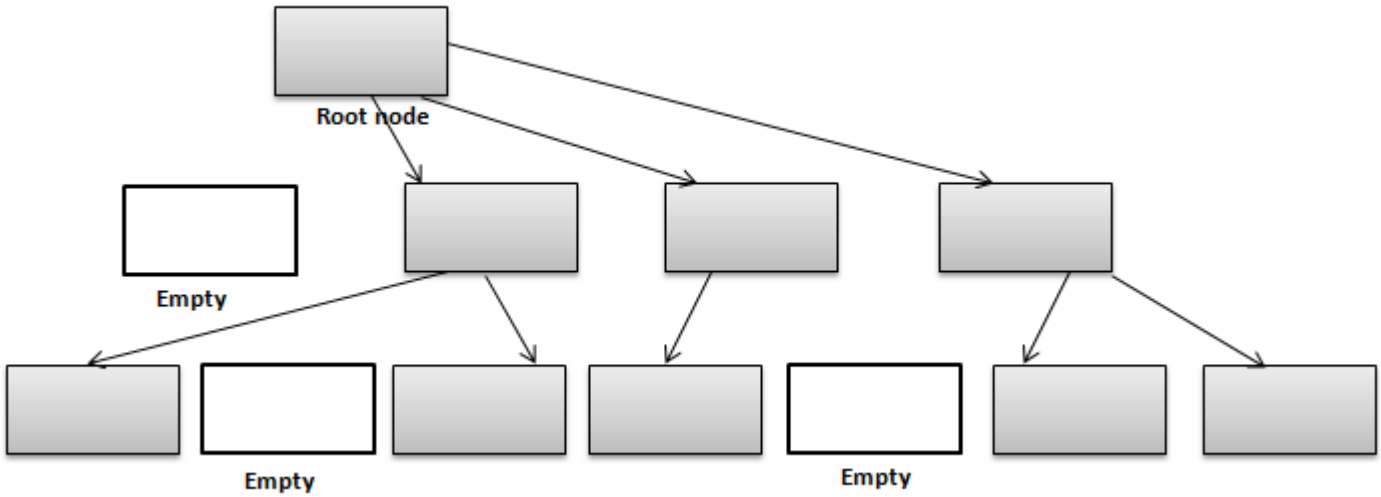


Figure 74: B-tree Structure

In the event of the deletion of a file or folder, there is a possibility of recovering the metadata of the file, (such as its name and attributes), as well as the actual data that the file consists of. **Active@ KillDisk**'s Wipe method clears out all of this free space in the system files.

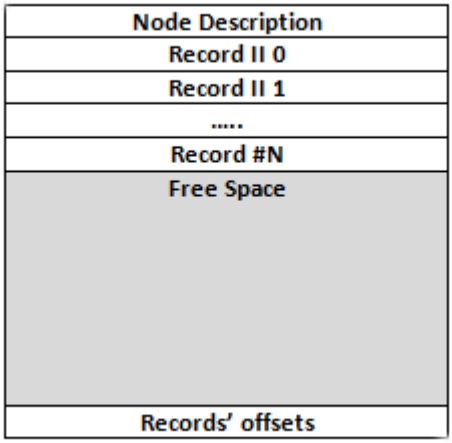


Figure 75: HFS+ System Table

Specifics of wiping Linux Ext2/Ext3/Ext4 file systems

A Linux Ext file system (Ext2/Ext3/Ext4) volume has a global descriptors table. Descriptors table records are called group descriptors and describe each blocks group. Each blocks group has an equal number of data blocks.

A data block is the smallest allocation unit: size vary from 1024 bytes to 4096 bytes. Each group descriptor has a blocks allocation bitmap. Each bit of the bitmap shows whether the block is allocated (1) or available (0). **Active@ KillDisk** software enumerates all groups, and for each and every block within the group on the volume checks the related bitmap to define its availability. If the Block is available, **Active@ KillDisk** wipes it using the method supplied by the user.

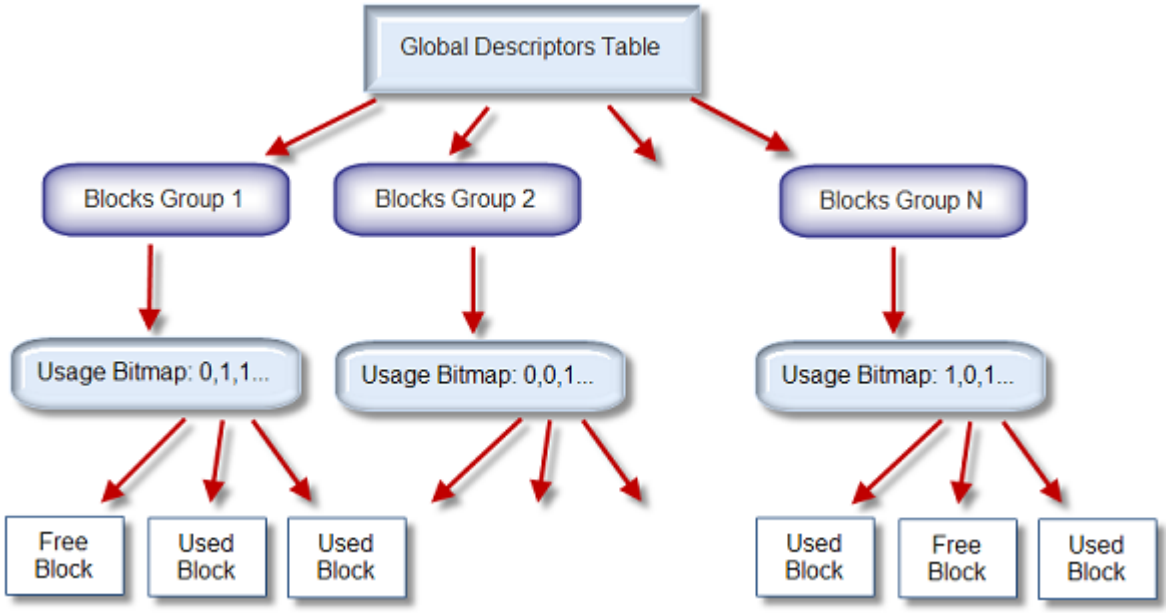


Figure 76: Ext2/Ext3/Ext4 Descriptors Table

Sanitization Types

NIST 800-88 international security standard (Guidelines for Media Sanitization) defines different types of sanitization.

Regarding sanitization, the principal concern is ensuring that data is not unintentionally released. Data is stored on media, which is connected to a system. Simply data sanitization applied to a representation of the data as stored on a specific media type.

When media is re-purposed or reaches end of life, the organization executes the system life cycle sanitization decision for the information on the media. For example, a mass-produced commercial software program contained on a DVD in an unopened package is unlikely to contain confidential data. Therefore, the decision may be made to simply dispose of the media without applying any sanitization technique. Alternatively, an organization is substantially more likely to decide that a hard drive from a system that processed Personally Identifiable Information (PII) needs sanitization prior to Disposal.

Disposal without sanitization should be considered only if information disclosure would have no impact on organizational mission, would not result in damage to organizational assets, and would not result in financial loss or harm to any individuals. The security categorization of the information, along with internal environmental factors, should drive the decisions on how to deal with the media. The key is to first think in terms of information confidentiality, then apply considerations based on media type. In organizations, information exists that is not associated with any categorized system. Sanitization is a process to render access to target data (the data subject to the sanitization technique) on the media infeasible for a given level of recovery effort. The level of effort applied when attempting to retrieve data may range widely. NIST

SP 800-88 Rev. 1 Guidelines for Media Sanitization *Clear*, *Purge*, and *Destroy* are actions that can be taken to sanitize media. The categories of sanitization are defined as follows:

Clear

Clear applies logical techniques to sanitize data in all user-addressable storage locations for protection against simple non-invasive data recovery techniques; typically applied through the standard Read and Write commands to the storage device, such as by rewriting with a new value or using a menu option to reset the device to the factory state (where rewriting is not supported).

For HDD/SSD/SCSI/USB media this means overwrite media by using organizationally approved and validated overwriting technologies/methods/tools. *The Clear pattern* should be at least a single write pass with a fixed data value, such as all zeros. Multiple write passes or more complex values may optionally be used.

Active@ KillDisk supports *Clear* sanitization type through the ***Disk Erase*** command for all R/W magnetic types of media, more than 20 international sanitation methods including custom patterns implemented and can be used.

Purge

Purge applies physical or logical techniques that render Target Data recovery infeasible using state of the art laboratory techniques.

For HDD/SSD/SCSI/USB media this means ATA SECURE ERASE UNIT, ATA CRYPTO SCRAMBLE EXT, ATA EXT OVERWRITE, ATA/SCSI SANITIZE and other low-level direct controller commands.

Active@ KillDisk supports *Purge* sanitization type through the *Secure Erase* command only for media types supporting ATA extensions.

Destroy

Destroy renders Target Data recovery infeasible using state of the art laboratory techniques and results in the subsequent inability to use the media for storage of data due to physical damages.

For HDD/SSD/SCSI media this means Shred, Disintegrate, Pulverize, or Incinerate by burning the device in a licensed incinerator.

It is suggested that the user categorize the information, assess the nature of the medium on which it is recorded, assess the risk to confidentiality, and determine the future plans for the media. Then, the organization can choose the appropriate type(s) of sanitization. The selected type(s) should be assessed as to cost, environmental impact, etc., and a decision should be made that best mitigates the risk to confidentiality and best satisfies other constraints imposed on the process.

Disk Erase performance

How fast erasing occurs? An actual erase speed depends on many factors:

- HDD/SSD/NVMe disk speed: RPM and SATA/SCSI/SAS/NVMe type - the most important factors
- Disk Controller speed: SAS (6 Gbps/12 Gbps), SATA III (6Gbps), (SATA II 3 Gbps), SATA I (1.5 Gbps)
- Computer overall performance (CPU, RAM) and workload (how many parallel erases occur)

For most modern computers and disks manufactured within last years SATA III standard is supported, so erase speed is limited by HDD throughput (disk write speed) only.

Our tests give the results: **10 GB per minute (in average) per pass** with decent computer configuration and disks with age of up to 5 years old.

For example, 2 TB Toshiba disk has been erased on Windows platform with one pass within 3 hours and 32 minutes, 14 TB Western Digital disk - within 18 hours 53 minutes.

The following snapshots are real-test certificates for erasing of:

1) **2 TB** Toshiba (manufactured in 2015) SATA III (6 GBps) 7200 rpm disk with One Pass Zeros and *US DoD 5220.22-M* (3 passes + verification) showing the average speed of **9 GB/min per pass**

Active@ KillDisk

ERASE CERTIFICATE

Erased by Active@ KillDisk 100% Guarantee

Disk Erase

Attributes

Erase Method: One Pass Zeros, 1 pass
Verification: No
Use Fingerprint: No
Initialize Disk: No

Disk Information

Name: PhysicalDrive1
Product Name: TOSHIBA DT01ACA200
Serial Number: X5G677ATS
Platform Name: \\.\PhysicalDrive1
Partitioning: RAW (Basic)
Size: 1.82 TB
Total Sectors: 3,907,029,168
Bytes per Sector: 512

Results

Erase Range: Whole disk
Name: Erasing PhysicalDrive1
Started at: 07/05/2020 10:04:27
Duration: 03:32:19
Errors: No Errors
Result: Erased

System Information

OS: Windows 10 (10.0) Professional 64-bit
Type: x64 (AMD or Intel)

Active@ KillDisk

ERASE CERTIFICATE

Erased by Active@ KillDisk 100% Guarantee

Disk Erase

Attributes

Erase Method: US DoD 5220.22-M, 3 passes
Verification: 1%
Use Fingerprint: No
Initialize Disk: No

Disk Information

Name: PhysicalDrive1
Product Name: TOSHIBA DT01ACA200
Serial Number: X5G677ATS
Platform Name: \\.\PhysicalDrive1
Size: 1.82 TB
Total Sectors: 3,907,029,168
Bytes per Sector: 512

Results

Erase Range: Whole disk
Name: Erasing PhysicalDrive1
Started at: 06/05/2020 17:52:12
Duration: 10:41:40
Errors: No Errors
Result: Erased

Erase Passes
Pass 1 (0x000000000000) - OK
Pass 2 (0xFFFFFFFFFFFF) - OK
Pass 3 (Random) - OK
Verification - passed OK

System Information

OS: Windows 10 Professional 64-bit
Type: x64 (AMD or Intel)

Hardware Information

Manufacturer: System manufacturer
Description: AT/AT COMPATIBLE
Logical Processors: 8
Memory: 15.8 GB
Name: System Product Name
System: x64-based PC
Physical Processors: 1

2) **14 TB** (Western Digital manufactured in 2019) SATA III (6 Gbps) 7200 rpm disk with One Pass Zeros and *US DoD 5220.22-M* (3 passes + 10% verification) showing the average speed of **12 GB/min per pass**

Active@ KillDisk

ERASE CERTIFICATE

Disk Erase

Attributes

Erase Method: **One Pass Zeros, 1 pass**
Verification: **No**
Use Fingerprint: **No**
Initialize Disk: **No**

Disk Information

Name: **PhysicalDrive1**
Product Name: **WDC WUH721414ALE6L4**
Serial Number: **Z2H2VXGT**
Platform Name: **\\.\PhysicalDrive1**
Size: **12.7 TB**
Total Sectors: **27,344,764,928**
Bytes per Sector: **512**

Results

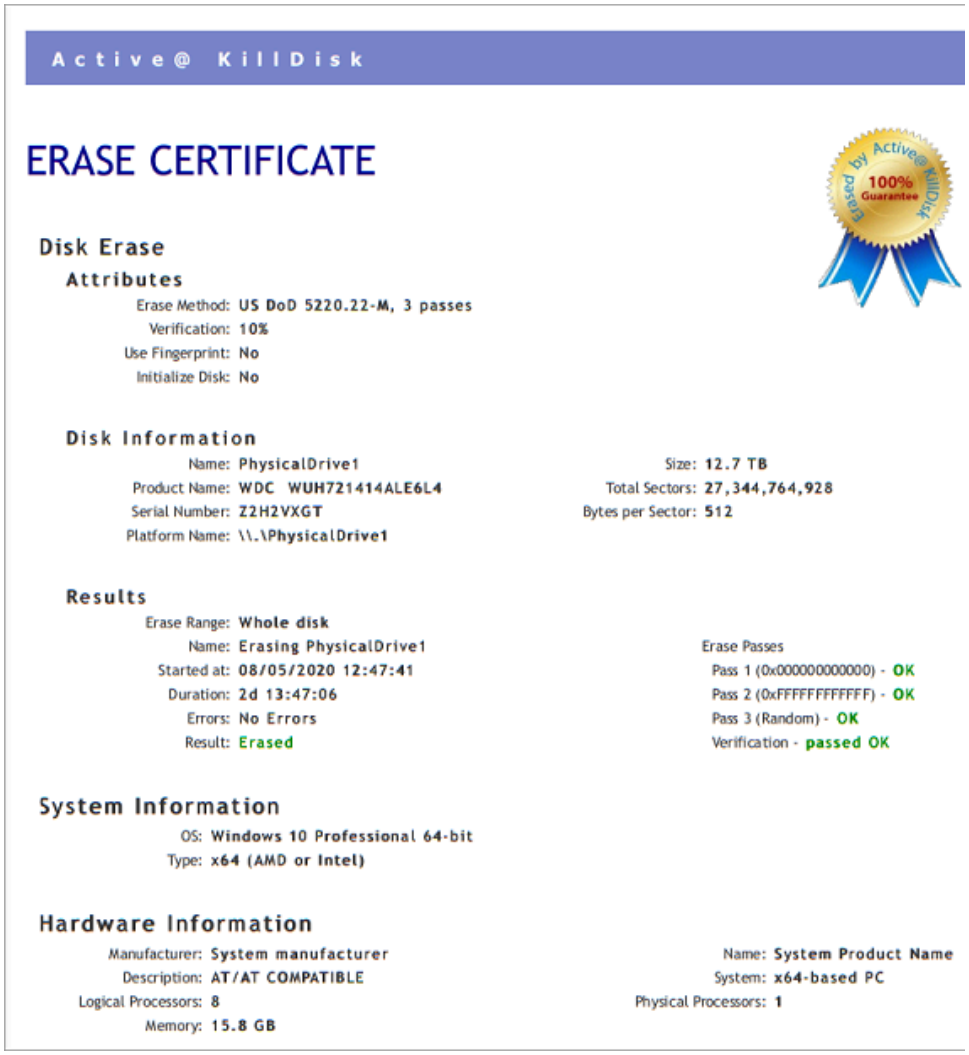
Erase Range: **Whole disk**
Name: **Erasing PhysicalDrive1**
Started at: **07/05/2020 17:48:54**
Duration: **18:53:08**
Errors: **No Errors**
Result: **Erased**

System Information

OS: **Windows 10 Professional 64-bit**
Type: **x64 (AMD or Intel)**

Hardware Information

Manufacturer: **System manufacturer**
Description: **AT/AT COMPATIBLE**
Logical Processors: **8**
Memory: **15.8 GB**
Name: **System Product Name**
System: **x64-based PC**
Physical Processors: **1**



Active@ KillDisk

ERASE CERTIFICATE

Erased by Active@ KillDisk 100% Guarantee

Disk Erase

Attributes

Erase Method: US DoD 5220.22-M, 3 passes
Verification: 10%
Use Fingerprint: No
Initialize Disk: No

Disk Information

Name: PhysicalDrive1
Product Name: WDC WUH721414ALE6L4
Serial Number: Z2H2VXGT
Platform Name: \\.\PhysicalDrive1
Size: 12.7 TB
Total Sectors: 27,344,764,928
Bytes per Sector: 512

Results

Erase Range: Whole disk
Name: Erasing PhysicalDrive1
Started at: 08/05/2020 12:47:41
Duration: 2d 13:47:06
Errors: No Errors
Result: Erased

Erase Passes
Pass 1 (0x000000000000) - OK
Pass 2 (0xFFFFFFFFFFFF) - OK
Pass 3 (Random) - OK
Verification - passed OK

System Information

OS: Windows 10 Professional 64-bit
Type: x64 (AMD or Intel)

Hardware Information

Manufacturer: System manufacturer
Description: AT/AT COMPATIBLE
Logical Processors: 8
Memory: 15.8 GB
Name: System Product Name
System: x64-based PC
Physical Processors: 1

Disk Hidden Zones

Active@ KillDisk is able to detect and reset Disk's Hidden Zones: HPA and DCO.

Host Protected Area

The Host Protected Area (*HPA*) is an area of a hard drive or solid-state drive that is not normally visible to an operating system. It was first introduced in the ATA-4 standard CXV (T13) in 2001.

How it works:

The IDE controller has registers that contain data that can be queried using ATA commands. The data returned gives information about the drive attached to the controller. There are three ATA commands involved in creating and using a host protected area. The commands are:

- IDENTIFY DEVICE
- SET MAX ADDRESS
- READ NATIVE MAX ADDRESS

Operating systems use the IDENTIFY DEVICE command to find out the addressable space of a hard drive. The IDENTIFY DEVICE command queries a particular register on the IDE controller to establish the size of a drive.

This register however can be changed using the SET MAX ADDRESS ATA command. If the value in the register is set to less than the actual hard drive size then effectively a host protected area is created. It is protected because the OS will work with only the value in the register that is returned by the IDENTIFY DEVICE command and thus will normally be unable to address the parts of the drive that lie within the HPA.

The HPA is useful only if other software or firmware (e. g. BIOS) is able to use it. Software and firmware that are able to use the HPA are referred to as 'HPA aware'. The ATA command that these entities use is called READ NATIVE MAX ADDRESS. This command accesses a register that contains the true size of the hard drive. To use the area, the controlling HPA-aware program changes the value of the register read by IDENTIFY DEVICE to that found in the register read by READ NATIVE MAX ADDRESS. When its operations are complete, the register read by IDENTIFY DEVICE is returned to its original fake value.

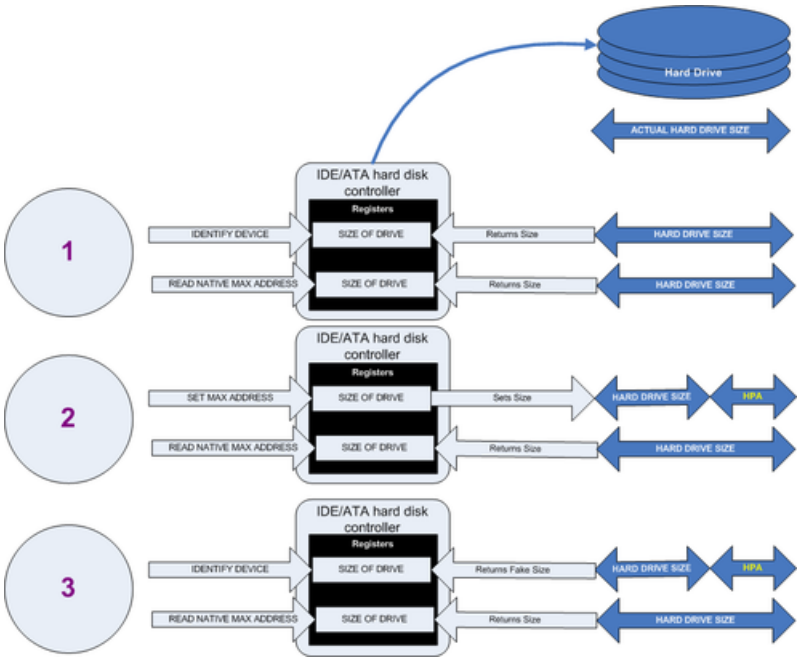


Figure 77: Creation of an HPA

The diagram shows how a host protected area (HPA) is created:

1. IDENTIFY DEVICE returns the true size of the hard drive. READ NATIVE MAX ADDRESS returns the true size of the hard drive.
2. SET MAX ADDRESS reduces the reported size of the hard drive. READ NATIVE MAX ADDRESS returns the true size of the hard drive. An HPA has been created.
3. IDENTIFY DEVICE returns the now fake size of the hard drive. READ NATIVE MAX ADDRESS returns the true size of the hard drive, the HPA is in existence.

Usage:

- At the time HPA was first implemented on hard-disk firmware, some BIOS had difficulty booting with large hard disks. An initial HPA could then be set (by some jumpers on the hard disk) to limit the number of cylinder to 4095 or 4096 so that older BIOS would start. It was then the job of the boot loader to reset the HPA so that the operating system would see the full hard-disk storage space.

- HPA can be used by various booting and diagnostic utilities, normally in conjunction with the BIOS. An example of this implementation is the Phoenix First BIOS, which uses Boot Engineering Extension Record (BEER) and Protected Area Run Time Interface Extension Services (PARTIES). Another example is the Gujin installer which can install the bootloader in BEER, naming that pseudo-partition /dev/hda0 or /dev/sdb0; then only cold boots (from power-down) will succeed because warm boots (from Ctrl-Alt-Delete) will not be able to read the HPA.
- Computer manufacturers may use the area to contain a preloaded OS for install and recovery purposes (instead of providing DVD or CD media).
- Dell notebooks hide Dell MediaDirect utility in HPA. IBM ThinkPad and LG notebooks hide system restore software in HPA.
- HPA is also used by various theft recovery and monitoring service vendors. For example, the laptop security firm Computrace use the HPA to load software that reports to their servers whenever the machine is booted on a network. HPA is useful to them because even when a stolen laptop has its hard drive formatted the HPA remains untouched.
- HPA can also be used to store data that is deemed illegal and is thus of interest to government and police.
- Some vendor-specific external drive enclosures (Maxtor) are known to use HPA to limit the capacity of unknown replacement hard drives installed into the enclosure. When this occurs, the drive may appear to be limited in size (e.g. 128 GB), which can look like a BIOS or dynamic drive overlay (DDO) problem. In this case, one must use software utilities (see below) that use READ NATIVE MAX ADDRESS and SET MAX ADDRESS to change the drive's reported size back to its native size, and avoid using the external enclosure again with the affected drive.
- Some rootkits hide in the HPA to avoid being detected by anti-rootkit and antivirus software.
- Some NSA exploits use the HPA for application persistence.

Device Configuration Overlay

Device Configuration Overlay (*DCO*) is a hidden area on many of today's hard disk drives (HDDs). Usually when information is stored in either the DCO or host protected area (HPA), it is not accessible by the BIOS, OS, or the user. However, certain tools can be used to modify the HPA or DCO. The system uses the IDENTIFY_DEVICE command to determine the supported features of a given hard drive, but the DCO can report to this command that supported features are nonexistent or that the drive is smaller than it actually is. To determine the actual size and features of a disk, the DEVICE_CONFIGURATION_IDENTIFY command is used, and the output of this command can be compared to the output of IDENTIFY_DEVICE to see if a DCO is present on a given hard drive. Most major tools will remove the DCO in order to fully image a hard drive, using the DEVICE_CONFIGURATION_RESET command. This permanently alters the disk, unlike with the (HPA), which can be temporarily removed for a power cycle.

Usage:

The Device Configuration Overlay (DCO), which was first introduced in the ATA-6 standard, "allows system vendors to purchase HDDs from different manufacturers with potentially different sizes, and then configure all HDDs to have the same number of sectors. An example of this would be using DCO to make an 80-gigabyte HDD appear as a 60-gigabyte HDD to both the (OS) and the BIOS.... Given the potential to place data in these hidden areas, this is an area of concern for computer forensics investigators. An additional issue for forensic investigators is imaging the HDD that has the HPA and/or DCO on it. While certain vendors claim that their tools are able to both properly detect and image the HPA, they are either silent on the handling of the DCO or indicate that this is beyond the capabilities of their tool.

Glossary

ANSI

It's a repertoire of character encoding that include (most of) the original 96 ASCII on page 205 character set, plus up to 128 additional characters.

ApFS

Apple File System is a proprietary file system developed and deployed by Apple Inc. for macOS Sierra (10.12.4) and later, iOS 10.3, tvOS 10.2, watchOS 3.2, and all versions of iPadOS. APFS is optimized for solid-state drive storage and supports encryption, snapshots, and increased data integrity, among other capabilities.

ASCII

An acronym for American Standard Code for Information Interchange, is a character encoding standard for electronic communication. ASCII codes represent text in computers, telecommunications equipment, and other devices. The ASCII standard is a seven-bit code with no parity guidelines, containing 128 code positions.

AT Attachment (ATA)

ATA is originally AT Attachment standard interface designed for connection storage devices such as hard disk drives, super floppy drives, optical disc drives, and tape drives in computers.

Integrated Drive Electronics (IDE) on page 210

Parallel ATA (PATA) on page 213

BCD

Boot **C**onfiguration **D**ata. Firmware-independent database for boot-time configuration data. It is used by Microsoft's new Windows Boot Manager and replaces the **boot.ini** that was used by NTLDR on page 212.

BIOS Settings

Basic **I**nput **O**utput **S**ubsystem is the program a personal computer's microprocessor uses to get the computer system started after you turn it on. It also manages data flow between the computer's operating system and attached devices such as the hard disk, video adapter, keyboard, mouse and printer. A typical method to access the BIOS settings screen is to press ***Delete*** / ***F1*** / ***F2*** / ***F8*** / ***F10*** or ***Esc*** during the boot sequence.

Blu-ray

Blu-ray (Blu-ray Disc or BD) is a digital optical disc data storage format designed to supersede the DVD format. The main application of Blu-ray is as a medium for video material and for the physical distribution of video games. The name refers to the blue laser used to read the disc, which allows information to be stored at a greater density than is possible with the longer-wavelength red laser used for DVDs, resulting in an increased capacity.

Boot partition

Name commonly used for the partition that contains the start-up files.

Boot Priority

BIOS settings allow you to run a boot sequence from a floppy drive, a hard drive, a CD/DVD/BD drive or a USB device. You may configure the order that your computer searches these physical devices for the boot sequence. The first device in the order list has the first boot priority. For example, to boot from a CD/DVD/BD drive instead of a hard drive, place the CD/DVD/BD drive ahead of the hard drive in priority.

Boot Record

The information about primary partitions and an extended partition contained in the Partition Table. Located in the physical disk's first sector. Each volume on the disk has its own Boot Record called Volume or Partition Boot Sector, the content is file system specific.

Master Boot Record (MBR) on page 132

Boot Sector

Part of a hard disc, floppy disc, or similar data storage device that contains code for bootstrapping programs (usually, but not necessarily, operating systems) stored in other parts of the disc. The boot sector continues the process of loading the operating system into computer memory. It can be either the Master Boot Record (MBR) on page 212 or the Partition Boot Sector on page 213 .

Master Boot Record (MBR) on page 132

BtrFS

A computer storage format that combines a file system based on the copy-on-write (COW) principle with a logical volume manager, developed together. It was created by Chris Mason in 2007 for use in Linux, and since November 2013, the file system's on-disk format has been declared stable in the Linux kernel. BtrFS is intended to address the lack of pooling, snapshots, checksums, and integral multi-device spanning in Linux file systems.

Compact Disc (CD)

The compact disc (CD) is a digital optical disc data storage format co-developed by Philips and Sony to store and play digital audio recordings. It employs the Compact Disc Digital Audio standard and is capable of holding of uncompressed stereo audio. In later years, the technology was adapted for computer data storage as CD-ROM and subsequently expanded into various writable and multimedia formats.

Compressed Cluster

When you set a file or folder property to compress data, the file or folder uses less disk space. While the size of the file is smaller, it must use a whole cluster in order to exist on the hard drive. As a result, compressed clusters contain File Slack Space on page 209. This space may contain residual confidential data from the file that previously occupied this space. **LSoft KillDisk** can wipe out the residual data without touching the existing data.

Comma-Separated Values (CSV) file

A comma-separated values (**CSV**) file is a delimited text file that uses a comma to separate values. Each line of the file is a data record. Each record consists of one or more fields, separated by commas. The use of the comma as a field separator is the source of the name for this file format. A CSV-file typically stores tabular data (numbers and text) in plain text, in which case each line will have the same number of fields.

Data Cluster

A cluster or allocation unit is a unit of disk space allocation for files and directories. To reduce the overhead of managing on-disk data structures, the file system does not allocate individual disk sectors by default, but contiguous groups of sectors, called clusters. A cluster is the smallest logical amount of disk space that can be allocated to hold a file. Storing small files on a file system with large clusters will therefore waste disk space; such wasted disk space is called slack space. For cluster sizes which are small versus the average file size, the wasted space per file will be statistically about half of the cluster size; for large cluster sizes, the wasted space will become greater. However, a larger cluster size reduces bookkeeping overhead and fragmentation, which may improve reading and writing speed overall. Typical cluster sizes range from 1 sector (512 B) to 128 sectors (64 Kb). The operating system keeps track of clusters in the hard disk's root records or MFT records, see Lost Cluster on page 211.

Data storage device

See physical device.

Device Configuration Overlay (DCO)

Device Configuration Overlay (*DCO*) is a hidden area on many of today's hard disk drives (HDDs). Usually when information is stored in either the DCO or host protected area (HPA), it is not accessible by the BIOS, OS, or the user.

Device Configuration Overlay on page 204

Deleted Boot Records

All disks and partitions start with a boot sector. For a damaged disk and volumes (where the location of the boot records known) the partition table can be reconstructed. The boot record contains a file system identifier.

Device Node

Device node in the Local System Devices list is a physical device containing logical drives. The first physical device on older versions of Operating Systems is named 80h, now more typical name is PhysicalDrive0.

Directory Entry

Basically the mapping of filename to its Inode on page 211. The user generally accesses the file by its name, however such filenames are not understood by the kernel. The kernel identifies a file using the inode which is unique for a file.

Disk geometry

Set of disk attributes that specify format, partitioning etc. of a disk.

Drive letter

Abstraction at the user level to distinguish one disk or partition from another. For example, the path C:\WINDOWS\represents a directory WINDOWS on the partition represented by C:.

Dynamic Disk

A dynamic storage made out of whole or part of physical disk to increase performance and reliability.

DVD

The DVD (digital video disc or digital versatile disc) is a digital optical disc data storage format. The medium can store any kind of digital data and has been widely used to store video programs, software and other computer files. DVDs offer significantly higher storage capacity than compact discs (CD) while having the same dimensions.

Big Endian/Little Endian

In computing, endianness is the order in which bytes within a word of digital data are transmitted over a data communication medium or addressed (by rising addresses) in computer memory, counting only byte significance compared to earliness. Endianness is primarily expressed as big-endian (BE) or little-endian (LE). CPU may read a digital word big end first, or little end first.

External SATA (eSATA)

External SATA is a SATA interface created for connecting external devices with support for the "hot-swap" mode.

SATA

SATA on page 216

Exclusive Access

Lock is applied to a partition for exclusive writing access. For example, while recovering deleted or damaged files or folders, the recovery application must have exclusive access to the target partition while recovering files. If another application or the operating system are using the target partition - the processes could interfere, so user/process must close all applications or system processes that may be using the target partition before locking it.

exFAT

Extensible File Allocation Table is a file system introduced by Microsoft in 2006 and optimized for flash memory such as USB flash drives and SD cards.

Ext

The extended file system was implemented in April 1992 as the first file system created specifically for the Linux kernel. Although EXT is not a specific file system name, it has been succeeded by EXT2, EXT3 and EXT4. It has metadata structure inspired by traditional Unix filesystem principles, and was designed by Rémy Card to overcome certain limitations of the MINIX file system. It was the first implementation that used the virtual file system (VFS), for which support was added in the Linux kernel in version 0.96c, and it could handle file systems up to 2 gigabytes (GB) in size.

Ext2

Second extended file system is a file system for the Linux kernel. It was initially designed by French software developer Rémy Card as a replacement for the extended file system (ext). Having been designed according to the same principles as the Berkeley Fast File System from BSD, it was the first commercial-grade filesystem for Linux. The canonical implementation of ext2 is the "ext2fs" filesystem driver in the Linux kernel. ext2 was the default filesystem in several Linux distributions, including Debian and Red Hat Linux.

Ext3

Third extended file system, is a journaled file system that is commonly used by the Linux kernel. It used to be the default file system for many popular Linux distributions. Stephen Tweedie first revealed that he was working on extending ext2 in Journaling the Linux ext2fs File system in a 1998 paper, and later in a February 1999 kernel mailing list posting. The file system was merged with the mainline Linux kernel in November 2001 from 2.4.15 onward. Its main advantage over ext2 is journaling, which improves reliability and eliminates the need to check the file system after an unclean shutdown.

Ext4

Fourth extended file system was initially a series of backward-compatible extensions to ext3, many of them originally developed by Cluster File Systems for the Lustre file system between 2003 and 2006, meant to extend storage limits and add other performance improvements. However, other Linux kernel developers opposed accepting extensions to ext3 for stability reasons, and proposed to fork the source code of ext3, rename it as ext4, and perform all the development there, without affecting existing ext3 users. This proposal was accepted, and on 28 June 2006, Theodore Ts'o, the ext3 maintainer, announced the new plan of development for ext4. A preliminary development version of ext4 was included in version 2.6.19 of the Linux kernel. On 11 October 2008, the patches that mark ext4 as stable code were merged in the Linux 2.6.28 source code repositories, denoting the end of the development phase and recommending ext4 adoption. Kernel 2.6.28, containing the ext4 file system, was finally released on 25 December 2008. On 15 January 2010, Google announced that it would upgrade its storage infrastructure from ext2 to ext4. On 14 December 2010, Google also announced it would use ext4, instead of YAFFS, on Android 2.3.

Extended Partition

A hard disk may contain only one extended partition; the extended partition can be subdivided into multiple logical partitions. In DOS/Windows systems, each logical partition may then be assigned an additional drive letter.

FAT (File Allocation Table)

Area that contains the records of every other file and directory in a FAT-formatted disk drive. The operating system needs this information to access the files. There are FAT32, FAT16 and exFAT versions. FAT file systems are still commonly found on flash disks and other memory cards and modules (including USB flash drives), as well as many portable and embedded devices. FAT is the standard file system for digital cameras per the DCF specification.

FAT32

A new version of the file system FAT16 which supported an increased number of possible clusters, but could reuse most of the existing code, so that the conventional memory footprint was increased by less than 5 KB under DOS. Cluster values are represented by 32-bit numbers, of which 28 bits are used to hold the cluster number.

File record

Each file record assigns a number, called a file index, that is used by later records to refer to the file described by this record. The described file might be the primary PL/I source file or an INCLUDEd file. Each file record specifies a literal index for the fully qualified name of the file. For an INCLUDEd file, each file record also contains the file index and source line number from where the INCLUDE request came. (For primary source files, these fields are zero.)

File Signature

Set of unique file properties, that allows recognize file type. File types are recognized by specific patterns that may serve as a reference for file recovery. When a file header is damaged, the type of file may be determined by examining patterns in the damaged file and comparing these patterns to known file type templates.

File Slack Space

The smallest file (and even an empty folder) takes up an entire cluster. A 10-byte file will take up 2,048 bytes if that is the cluster size. File slack space is the unused portion of a cluster. This space may contain residual confidential data from the file that previously occupied this space. **LSoft KillDisk** can wipe out the residual data without touching the existing data.

File system

Method in which files are named and where they are placed logically for storage and retrieval in a computer. Under scope of this document, one of the Microsoft Windows file systems, such as FAT12, FAT16, FAT32 and NTFS.

File Table

A specialized table with a predefined schema that stores FILESTREAM data, as well as file and directory hierarchy information and file attributes.

FTP

File **T**ransfer **P**rotocol. This is a standard network protocol used for the transfer of computer files between a Client and Server on a computer network. FTP is built on a client-server model architecture using separate control and data connections between the client and the server. FTP users may authenticate themselves with a clear-text sign-in protocol, normally in the form of a username and password, but can connect anonymously if the server is configured to allow it. For secure transmission that protects the username and password, and encrypts the content, FTP is often secured with SSL/TLS (FTPS) or replaced with SSH File Transfer Protocol (SFTP). The first FTP client applications were command-line programs developed before operating systems had graphical user interfaces, and are still shipped with most Windows, Unix, and Linux operating systems. Many FTP clients and automation utilities have since been developed for desktops, servers, mobile devices, and hardware, and FTP has been incorporated into productivity applications, such as HTML editors.

Free Cluster

A cluster that is not occupied by a file. This space may contain residual confidential data from the file that previously occupied this space. **LSoft KillDisk** can wipe out the residual data.

FreeDOS

A free operating system for PC compatible computers. It intends to provide a complete DOS-compatible environment for running legacy software and supporting embedded systems. FreeDOS can be booted from a floppy disk or USB flash drive. It is designed to run well under virtualization or x86 emulation. Unlike most versions of MS-DOS, FreeDOS is composed of free and open-source software, licensed under the terms of the GNU General Public License.

GUID Partition Table (GPT)

A standard for the layout of partition tables of a physical computer storage device, such as a hard disk drive or solid-state drive, using universally unique identifiers (UUIDs), which are also known as globally unique identifiers (GUIDs). Forming a part of the Unified Extensible Firmware Interface (UEFI) standard (Unified EFI Forum-proposed replacement for the PC BIOS), it is nevertheless also used for some BIOSs, because of the limitations of MBR partition tables, which use 32 bits for logical block addressing (LBA) of traditional 512-byte disk sectors. GPT provides a more flexible mechanism for partitioning disks than the older MBR partitioning scheme that was common to PCs.

Hard disk drive(HDD)

A hard disk drive (HDD), hard disk, hard drive, or fixed disk[a] is an electro-mechanical data storage device that stores and retrieves digital data using magnetic storage with one or more rigid rapidly rotating platters coated with magnetic material.

HEX

Hexadecimal (also base-16 or simply hex) numeral system is a positional numeral system that represents numbers using a radix (base) of sixteen. Hexadecimal uses sixteen distinct symbols, most often the symbols "0"–"9" to represent values 0 to 9 and "A"–"F" (or "a"–"f") to represent values from ten to fifteen.

HFS+

HFS Plus (also known as Mac OS Extended or HFS Extended) is a journaling file system developed by Apple Inc. It replaced the Hierarchical File System (HFS) as the primary file system of Apple computers with the 1998 release of Mac OS 8.1. HFS+ continued as the primary Mac OS X file system until it was itself replaced with the Apple File System (APFS), released with macOS High Sierra in 2017. HFS+ is also one of the formats supported by the iPod digital music player. Compared to HFS, HFS+ supports much larger files (block addresses are 32-bit length instead of 16-bit) and using Unicode (instead of Mac OS Roman or any of several other character sets) for naming items. Like HFS, HFS Plus uses B-trees to store most volume metadata, but unlike most file systems that support hard links, HFS Plus supports hard links to directories. HFS Plus permits filenames up to 255 characters in length, and n-forked files. HFS Plus also uses a full 32-bit allocation mapping table rather than HFS's 16 bits, improving the use of space on large disks.

Host Protected Area (HPA)

The Host Protected Area (*HPA*) is an area of a hard drive or solid-state drive that is not normally visible to an operating system. It was first introduced in the ATA-4 standard CXV (T13) in 2001.

Host Protected Area on page 202

Integrated Drive Electronics (IDE)

Integrated Drive Electronics (IDE), originally AT Attachment (ATA) on page 205, also known as Parallel ATA (PATA), is a standard interface designed for IBM PC-compatible computers. The connection is used for storage devices such as hard disk drives, super floppy drives, optical disc drives, and tape drives in computers. It uses the underlying AT Attachment (ATA) and AT Attachment Packet Interface (ATAPI) standards. The Parallel ATA standard is the result of a long history of incremental technical development, which began with the original AT Attachment interface, developed for use in early PC AT equipment. The ATA interface itself evolved in several stages from Western Digital's original Integrated Drive Electronics (IDE) interface. As a result, many near-synonyms for ATA/ATAPI and its previous incarnations are still in

common informal use, in particular Extended IDE (EIDE) and Ultra ATA (UATA). After the introduction of SATA in 2003, the original ATA was renamed to Parallel ATA, or PATA for short.

PATA
Parallel ATA (PATA) on page 213

Inode

Index node is a data structure in a Unix-style file system that describes a file-system object such as a file or a directory. Each inode stores the attributes and disk block locations of the object's data. File-system object attributes may include metadata (times of last change, access, modification), as well as owner and permission data. A directory is a list of inodes with their assigned names. The list includes an entry for itself, its parent, and each of its children.

iSCSI

Internet **S**mall **C**omputer **S**ystems **I**nterface. iSCSI is a transport layer protocol that works on top of the Transport Control Protocol (TCP). It enables block-level SCSI data transport between the iSCSI initiator and the storage target over TCP/IP networks.

ISO

An International Organization for Standardization ISO-9660 file system is a standard CD-ROM file system that allows you to read the same CD-ROM whether you're on a PC, Mac, or other major computer platform. Disk images of ISO-9660 file systems (ISO images) are a common way to electronically transfer the contents of CD-ROMs. They often have the file name extension .ISO (though not necessarily), and are commonly referred to as "ISO".

JFS

Journaled File System is a 64-bit journaling file system created by IBM. There are versions for AIX, OS/2, eComStation, ArcaOS and Linux operating systems. The latter is available as free software under the terms of the GNU General Public License (GPL). HP-UX has another, different filesystem named JFS that is actually an OEM version of Veritas Software's VxFS. In the AIX operating system, two generations of JFS exist, which are called JFS (JFS1) and JFS2 respectively. IBM's JFS was originally designed for 32-bit systems. JFS2 was designed for 64-bit systems. In other operating systems, such as OS/2 and Linux, only the second generation exists and is called simply JFS.

LDM

The Logical Disk Manager is an implementation of a logical volume manager for Microsoft Windows NT, developed by Microsoft and Veritas Software. It was introduced with the Windows 2000 operating system, and is supported in Windows XP, Windows Server 2003, Windows Vista, Windows 7, Windows 8, Windows 10 and Windows 11. The MMC-based Disk Management snap-in (diskmgmt.msc) hosts the Logical Disk Manager. On Windows 8 and Windows Server 2012, Microsoft deprecated LDM in favor of Storage Spaces. Logical Disk Manager enables disk volumes to be dynamic, in contrast to the standard basic volume format. Basic volumes and dynamic volumes differ in their ability to extend storage beyond one physical disk. Basic partitions are restricted to a fixed size on one physical disk. Dynamic volumes can be enlarged to include more free space - either from the same disk or another physical disk.

Logical drive

Partitioned space on a physical device.

Lost Cluster

A cluster that has an assigned number in the file allocation table, even though it is not assigned to any file. You can free up disk space by reassigning lost clusters. In DOS and Windows you can find lost clusters with the **ScanDisk** utility.

M.2

M.2 formerly known as the Next Generation Form Factor (NGFF), is a specification for internally mounted computer expansion cards and connectors. It was developed to replace the older Mini SATA (mSATA) and Mini PCIe (mPCIe) standards. M.2 supports a variety of module sizes and interface types, offering greater flexibility for modern devices. It is widely used in compact systems such as ultrabooks and tablet computers, particularly for solid-state drives (SSDs), due to its smaller size and higher performance compared to mSATA. The M.2 connector can provide multiple interface options, including up to four lanes of PCI Express, as well as Serial ATA 3.0 and USB 3.0. The supported interfaces vary depending on the device and host implementation. M.2 modules and slots use different "keying" notches to indicate supported interfaces and to prevent incompatible installations. For storage devices, M.2 supports both the older Advanced Host Controller Interface (AHCI) and the newer NVM Express (NVMe) protocols. AHCI provides compatibility with legacy SATA-based systems and operating systems, while NVMe is designed for high-speed SSDs and allows for much faster performance by supporting multiple simultaneous I/O operations.

Master Boot Record (MBR)

All disks start with a boot sector. When you start the computer, the code in the MBR executes before the operating system is started. The location of the MBR is always track (cylinder) 0, side (head) 0, and sector 1. The MBR contains a file system identifier.

MFT or MFT records (Master File Table)

File that contains the records of every other file and directory in an NTFS-formatted hard disk drive. The operating system needs this information to access the files.

Named Streams

#unique_252 supports multiple data streams where the stream name identifies a new data attribute on the file. A handle can be opened to each data stream. A data stream, then, is a unique set of file attributes. Streams have separate opportunistic locks, file locks, and sizes, but common permissions.

Node

A structure which may contain data and connections to other nodes, sometimes called edges or links. Each node in a tree has zero or more child nodes, which are below it in the tree (by convention, trees are drawn with descendants going downwards). A node that has a child is called the child's parent node (or superior). All nodes have exactly one parent, except the topmost root node, which has none. A node might have many ancestor nodes, such as the parent's parent. Child nodes with the same parent are sibling nodes. Typically siblings have an order, with the first one conventionally drawn on the left. Some definitions allow a tree to have no nodes at all, in which case it is called empty.

NTFS

New **T**echnology **F**ile **S**ystem (developed by Microsoft) is the file system that the Windows NT operating system uses for storing and retrieving files on a hard disk. NTFS is the Windows NT equivalent of the Windows 95 file allocation table (FAT (File Allocation Table) on page 208) and the OS/2 High Performance File System (HPFS). All the latest Windows Operating Systems (Windows Vista, Windows 7, Windows 10) still use NTFS as a default file system.

NTLDR

Aka NT loader is the boot loader for all releases of Windows NT operating system up to and including Windows XP and Windows Server 2003. NTLDR is typically run from the primary hard disk drive, but it can also run from portable storage devices such as a CD/DVD or USB flash drive.

Non-Volatile Memory Express (NVMe)

NVM Express (NVMe) or Non-Volatile Memory Host Controller Interface Specification (NVMHCIS) is an open, logical-device interface specification for accessing a computer's non-volatile storage media usually attached via the PCI Express bus. The initial NVM stands for non-volatile memory, which is often NAND

flash memory that comes in several physical form factors, including solid-state drives (SSDs), PCIe add-in cards, and M.2 cards, the successor to mSATA cards. NVM Express, as a logical-device interface, has been designed to capitalize on the low latency and internal parallelism of solid-state storage devices.

Offset

An offset denotes the number of address locations added to a base address in order to get to a specific absolute address. In this (original) meaning of offset, only the basic address unit, usually the 8-bit byte, is used to specify the offset's size. In this context an offset is sometimes called a relative address.

OpenSUSE

A Linux distribution. It is widely used throughout the world. The focus of its development is creating usable open-source tools for software developers and system administrators, while providing a user-friendly desktop and feature-rich server environment.

Partition

Hard disk's storage space divided into independent parts. Each partition can behave like a separate disk drive.

Partition Boot Sector

On NTFS on page 212 or FAT (File Allocation Table) on page 208 file systems, the partition boot sector is a small program that is executed when the operating system tries to access a particular partition. On personal computers, the Master Boot Record (MBR) on page 212 uses the partition boot sector on the system partition to determine file system type, cluster size, etc., and to load the operating system kernel files. Partition boot sector is usually the first sector of the partition.

Partition scheme

Partition schemes define how data is stored on the file system. The scheme is important because it determines how the data is queried.

Partition table

A table that contains data of partitions on a disk, divided into segments, called partitions. In this table, partitions data or a map of partitions is stored and organized for users to comprehend. The common location of such data used to be the first sector when Master Boot Record (MBR) on page 212 was the primary partitioning scheme in PCs. Nowadays, other more advanced formats divide a disk drive into partitions, like GUID Partition Table (GPT) on page 210 or Apple partition map (APM). As for GPT, this partitioning scheme is saved in sector 2 and keeps sector 1 empty for MBR data for backward compatibility.

Parallel ATA (PATA)

Parallel ATA (PATA), originally AT Attachment, also known as Integrated Drive Electronics (IDE)

Integrated Drive Electronics (IDE) on page 210

Physical device

Device for storing data, that can be connected internally (Hard Drive) or externally (USB Flash card, USB Hard Drive).

Physical device geometry

See Disk geometry on page 207.

PXE

Preboot E**X**ecution **E**nvironment. In computing the Preboot Execution Environment specification describes a standardized client-server environment that boots a software assembly, retrieved from a network, on PXE-

enabled clients. On the client side it requires only a PXE-capable network interface controller, and uses a small set of industry-standard network protocols such as DHCP and TFTP.

RAID

RAID ("**R**edundant **A**rray of **I**nexpensive **D**isks" or "**R**edundant **A**rray of **I**ndependent **D**isks") is a data storage virtualization technology that combines multiple physical disk drive components into one or more logical units for the purposes of data redundancy, performance improvement, or both. Data is distributed across the drives in one of several ways, referred to as RAID levels, depending on the required level of redundancy and performance. The different schemes, or data distribution layouts, are named by the word "RAID" followed by a number, for example RAID ***0*** or RAID ***1***. Each scheme, or *RAID* level, provides a different balance among the key goals: reliability, availability, performance, and capacity. RAID levels greater than RAID 0 provide protection against unrecoverable sector read errors, as well as against failures of whole physical drives.

RAID 0

RAID 0 consists of striping, but no mirroring or parity. Compared to a spanned volume, the capacity of a RAID 0 volume is the same; it is the sum of the capacities of the drives in the set. But because striping distributes the contents of each file among all drives in the set, the failure of any drive causes the entire RAID 0 volume and all files to be lost. In comparison, a spanned volume preserves the files on the unfailing drives. The benefit of RAID 0 is that the throughput of read and write operations to any file is multiplied by the number of drives because, unlike spanned volumes, reads and writes are done concurrently. The cost is increased vulnerability to drive failures—since any drive in a RAID 0 setup failing causes the entire volume to be lost, the average failure rate of the volume rises with the number of attached drives.

RAID 1

RAID 1 consists of data mirroring, without parity or striping. Data is written identically to two or more drives, thereby producing a "mirrored set" of drives. Thus, any read request can be serviced by any drive in the set. If a request is broadcast to every drive in the set, it can be serviced by the drive that accesses the data first (depending on its seek time and rotational latency), improving performance. Sustained read throughput, if the controller or software is optimized for it, approaches the sum of throughputs of every drive in the set, just as for RAID 0. Actual read throughput of most RAID 1 implementations is slower than the fastest drive. Write throughput is always slower because every drive must be updated, and the slowest drive limits the write performance. The array continues to operate as long as at least one drive is functioning.

RAID 2

RAID 2 consists of bit-level striping with dedicated Hamming-code parity. All disk spindle rotation is synchronized and data is striped such that each sequential bit is on a different drive. Hamming-code parity is calculated across corresponding bits and stored on at least one parity drive. This level is of historical significance only; although it was used on some early machines (for example, the Thinking Machines CM-2), as of 2014 it is not used by any commercially available system.

RAID 3

RAID 3 consists of byte-level striping with dedicated parity. All disk spindle rotation is synchronized and data is striped such that each sequential byte is on a different drive. Parity is calculated across corresponding bytes and stored on a dedicated parity drive. Although implementations exist, RAID 3 is not commonly used in practice.

RAID 4

RAID 4 consists of block-level striping with dedicated parity. This level was previously used by NetApp, but has now been largely replaced by a proprietary implementation of RAID 4 with two parity disks, called RAID-DP. The main advantage of RAID 4 over RAID 2 and 3 is I/O parallelism: in RAID 2 and 3, a single read I/O operation requires reading

the whole group of data drives, while in RAID 4 one I/O read operation does not have to spread across all data drives. As a result, more I/O operations can be executed in parallel, improving the performance of small transfers.

RAID 5

RAID 5 consists of block-level striping with distributed parity. Unlike RAID 4, parity information is distributed among the drives, requiring all drives but one to be present to operate. Upon failure of a single drive, subsequent reads can be calculated from the distributed parity such that no data is lost. RAID 5 requires at least three disks. Like all single-parity concepts, large RAID 5 implementations are susceptible to system failures because of trends regarding array rebuild time and the chance of drive failure during rebuild. Rebuilding an array requires reading all data from all disks, opening a chance for a second drive failure and the loss of the entire array.

RAID 6

RAID 6 consists of block-level striping with double distributed parity. Double parity provides fault tolerance up to two failed drives. This makes larger RAID groups more practical, especially for high-availability systems, as large-capacity drives take longer to restore. RAID 6 requires a minimum of four disks. As with RAID 5, a single drive failure results in reduced performance of the entire array until the failed drive has been replaced. With a RAID 6 array, using drives from multiple sources and manufacturers, it is possible to mitigate most of the problems associated with RAID 5. The larger the drive capacities and the larger the array size, the more important it becomes to choose RAID 6 instead of RAID 5. RAID 10 (see Nested RAID levels) also minimizes these problems

RAS

Remote **A**ccess **S**ervice. Is any combination of hardware and software to enable the remote access tools or information that typically reside on a network of IT devices. A remote access service connects a client to a host computer, known as a remote access server. The most common approach to this service is remote control of a computer by using another device which needs internet or any other network connection.

ReFS

Resilient File System, codenamed "Protogon", is a Microsoft proprietary file system introduced with Windows Server 2012 with the intent of becoming the "next generation" file system after NTFS. The key design advantages of ReFS include automatic integrity checking and data scrubbing, elimination of the need for running chkdsk, protection against data degradation, built-in handling of hard disk drive failure and redundancy, integration of RAID functionality, a switch to copy/allocate on write for data and metadata updates, handling of very long paths and filenames, and storage virtualization and pooling, including almost arbitrarily sized logical volumes (unrelated to the physical sizes of the used drives).

Registry hive

Highest level of organization in the Windows registry. It is a logical group of keys, subkeys, and values in the registry that has a set of supporting files loaded into memory when Windows is started or an user logs in.

Root directory

This is a Directory Table that stores information about the files and directories located in the root directory. It is only used with FAT12 and FAT16, and imposes on the root directory a fixed maximum size which is pre-allocated at creation of this volume. FAT32 stores the root directory in the Data Region, along with files and other directories, allowing it to grow without such a constraint. Thus, for FAT32, the Data Region starts here.

Root records

Used in FAT file system. A table that contains the records of every other file and directory in a FAT-formatted hard disk drive. The operating system needs this information to access the files. There are FAT32, FAT16 and FAT versions.

SAM

Security **A**ccount **M**anager. Database file that stores users' passwords in a hashed format. Since a hash function is one-way, this provides some measure of security for the storage of the passwords. It can be used to authenticate local and remote users. Beginning with Windows 2000 SP4, Active Directory authenticates remote users.

SAS

Serial Attached SCSI (SAS) is a point-to-point serial protocol that moves data to and from computer-storage devices such as hard disk drives, solid-state drives and tape drives. SAS replaces the older Parallel SCSI. SAS uses the standard SCSI command set.

SATA

SATA (Serial AT Attachment)[a][2] is a computer bus interface that connects host bus adapters to mass storage devices such as hard disk drives, optical drives, and solid-state drives. Serial ATA succeeded the earlier Parallel ATA (PATA) standard to become the predominant interface for storage devices.

SCSI

Small **C**omputer **S**ystem **I**nterface. A set of standards for physically connecting and transferring data between computers and peripheral devices. The SCSI standards define commands, protocols, electrical, optical and logical interfaces. SCSI is most commonly used for hard disk drives and tape drives, but it can connect a wide range of other devices, including scanners and CD drives, although not all controllers can handle all devices. The SCSI standard defines command sets for specific peripheral device types; the presence of "unknown" as one of these types means that in theory it can be used as an interface to almost any device, but the standard is highly pragmatic and addressed toward commercial requirements.

Secure Digital (SD) card

The SD card is a proprietary, non-volatile, flash memory card format developed by the SD Association (SDA). They come in three physical forms: the full-size SD, the smaller miniSD (now obsolete), and the smallest, microSD. Owing to their compact form factor, SD cards have been widely adopted in a variety of portable consumer electronics, including digital cameras, camcorders, video game consoles, mobile phones, action cameras, and camera drones.

Sector

The smallest unit that can be accessed on a disk. Typically sector size is 512 or 4096 bytes.

Secure Erase (SSD)

The ATA Secure Erase command is designed to remove all user data from a drive. With an SSD without integrated encryption, this command will put the drive back to its original out-of-box state. This will initially restore its performance to the highest possible level and the best (lowest number) possible write amplification, but as soon as the drive starts garbage collecting again the performance and write amplification will start returning to the former levels. Drives which encrypt all writes on the fly can implement ATA Secure Erase in another way. They simply zeroize and generate a new random encryption key each time a secure erase is done. In this way the old data cannot be read anymore, as it cannot be decrypted. Some drives with an integrated encryption will physically clear all blocks after that as well, while other drives may require a TRIM command to be sent to the drive to put the drive back to its original out-of-box state (as otherwise their performance may not be maximized).

Secure Erase (Frozen State)

SSD disk is blocked (frozen) by BIOS. The reasons can differ. Modern ATA hard drives and SSDs offer security options that help user to control access and reliably destroy data if necessary. Brand new HDD or SSD from a store have all the security features initially disabled... BIOS of many motherboards run the SECURITY_FREEZE_LOCK ATA command when booting to provide protection against manipulation.

Span Array

A series of dynamic drives linked together to make one contiguous spanned volume.

S.M.A.R.T.

Self-**M**onitoring, **A**nalysis and **R**eporting **T**echnology; often written as *SMART* is a monitoring system included in computer hard disk drives (HDDs), solid-state drives (SSDs) and embedded MultiMediaCards (eMMC) drives. Its primary function is to detect and report various indicators of drive reliability with the intent of anticipating imminent hardware failures. When SMART data indicates a possible imminent drive failure, software running on the host system may notify the user so preventative action can be taken to prevent data loss and the failing drive can be replaced and data integrity maintained.

Solid-state drive (SSD)

A solid-state drive (SSD) is a type of solid-state storage device that uses integrated circuits to store data persistently. It is sometimes called semiconductor storage device, solid-state device, or solid-state disk.

Superblock

A segment of metadata describing the file system on a block device, as in the Unix File System

System partition

Name commonly used for the partition that contains the operating system files.

Templates (Patterns)

File types are recognized by specific patterns that may serve as a reference for file recovery. When a file header is damaged, the type of file may be determined by examining patterns in the damaged file and comparing these patterns to known file type templates. This same pattern-matching process can be applied to deleted or damaged partitions. Using FAT (File Allocation Table) on page 208 or NTFS on page 212 templates, recovery software can assume that a particular sector is a FAT or NTFS boot sector because parts of it match a known pattern.

Tiny Core Linux

A minimal Linux kernel based operating system focusing on providing a base system functionality. The distribution is notable for its small size (11 to 16 MB) and minimalism; additional functions are provided by extensions. Tiny Core Linux is free and open source software and is licensed under the GNU General Public License version 2.

Track

Tracks are concentric circles around the disk and the sectors are segments within each circle.

U.2

U.2 formerly known as SFF-8639, is a computer interface standard used to connect solid-state drives (SSDs) to a computer. It defines the physical connector, electrical characteristics, and supported communication protocols. U.2 was developed for the enterprise storage market and is designed to support multiple types of drives, including those using PCI Express (typically with NVM Express), as well as SAS and SATA. The interface supports up to four PCIe lanes and two SATA lanes, enabling high data transfer rates while maintaining compatibility with existing drive technologies.

UEFI

Unified **E**xtensible **F**irmware **I**nterface is a specification for a software program that connects a computer's firmware to its operating system (OS). UEFI is expected to eventually replace BIOS. Like BIOS, UEFI is installed at the time of manufacturing and is the first program that runs when a computer is turned on.

UFS

The Unix file system is a family of file systems supported by many Unix and Unix-like operating systems. It is a distant descendant of the original filesystem used by Version 7 Unix.

Unallocated Space

Space on a hard disk where no partition exists. A partition may have been deleted or damaged or a partition may not have been created.

Unicode

Text encoding standard maintained by the Unicode Consortium designed to support the use of text in all of the world's writing systems that can be digitized.

Unused Space in MFT-records

Applicable to NTFS file system on Windows. The performance of the computer system depends a lot on the performance of the MFT. When you delete files, the MFT entry for that file is not deleted, it is marked as deleted. This is called unused space in the MFT. If unused space is not removed from the MFT, the size of the table could grow to a point where it becomes fragmented, affecting the performance of the MFT and possibly the performance of the computer. This space may also contain residual confidential data (file names, file attributes, resident file data) from the files that previously occupied these spaces. **Active@ KillDisk** can wipe out the residual data without touching the existing data.

USB flash drive

A flash drive (also thumb drive, memory stick, and pen drive/pendrive) is a data storage device that includes flash memory with an integrated USB interface. A typical USB drive is removable, rewritable, and smaller than an optical disc, and usually weighs less than 30 g (1 oz).

Virtual Disk Array

Software layer that sits above assembled physical disks that were part of a hardware RAID system.

Virtual partition

A virtual copy of a volume (logical drive) using a defined geometry that emulates a real logical drive or partition.

Virtual disk

A virtual copy of a physical disk using a defined disk geometry that uses real physical disk as a source but access it.

Virtual RAID array

See Virtual Disk Array on page 218

Volume

A fixed amount of storage on a hard disk. A physical device may contain a number of volumes. It is also possible for a single volume to span to a number of physical devices.

Volume boot record

First sector of a data storage device that has not been partitioned, or the first sector of an individual partition on a data storage device that has been partitioned. It contains code to load and invoke the operating system (or other standalone program) installed on that device or within that partition.

Volume Header

It contains information about the volume as a whole, including the location of other key structures in the volume. The volume header is always located at 1024 bytes from the start of the volume. A copy of the volume header, the alternate volume header, is stored starting 1024 bytes before the end of the volume.

Volume Shadow Copy

Shadow Copy (also known as Volume Snapshot Service, Volume Shadow Copy Service or VSS) is a technology included in Microsoft Windows that can create backup copies or snapshots of computer files or volumes, even when they are in use. It is implemented as a Windows service called the Volume Shadow Copy service.

Windows System Caching

Windows reserves a specified amount of volatile memory for file system operations. This is done in RAM because it is the quickest way to do these repetitive tasks.

Windows System Records

The Windows logs keeps track of almost everything that happens in Windows OS. This enhances performance of the computer when doing repetitive tasks. Over time, these records can take up a lot of space.

WinPE

WinPE is a compact Windows-based operating system used as a recovery environment to install, deploy, and repair Windows Desktop Editions, Windows Server, and other Windows operating systems. After boot to WinPE, user can:

- Set up a hard drive before installing Windows.
- Install Windows by using apps or scripts from a network or a local drive.
- Capture and apply Windows images.
- Modify the Windows operating system while it's not running.
- Set up automatic recovery tools.
- Recover data from unbootable devices.
- Add a custom shell or GUI to automate these kinds of tasks.

XFS

A high-performance 64-bit journaling file system created by Silicon Graphics, Inc (SGI) in 1993. It was the default file system in SGI's IRIX operating system starting with its version 5.3. XFS was ported to the Linux kernel in 2001; as of June 2014, XFS is supported by most Linux distributions; Red Hat Enterprise Linux uses it as its default file system. XFS excels in the execution of parallel input/output (I/O) operations due to its design, which is based on allocation groups (a type of subdivision of the physical volumes in which XFS is used- also shortened to AGs). Because of this, XFS enables extreme scalability of I/O threads, file system bandwidth, and size of files and of the file system itself when spanning multiple physical storage devices. XFS ensures the consistency of data by employing metadata journaling and supporting write barriers. Space allocation is performed via extents with data structures stored in B+ trees, improving the overall performance of the file system, especially when handling large files. Delayed allocation assists in the prevention of file system fragmentation; online defragmentation is also supported.

Uninstall Active@ KillDisk

How to uninstall Active@ KillDisk.

Active@ KillDisk software comes with a standard installer\uninstaller accessible from the Control Panel.

To uninstall the software:

1. Open Control Panel;
2. Navigate ***Programs & Features*** > ***Uninstall*** or change a program;
3. Select **Active@ KillDisk** section and click ***Uninstall*** or just double click it;
4. Click ***Yes*** to confirm uninstall process;