Virtual Hard Disk Image Format Specification

This specification is the work of Microsoft Corporation, with clarifications and corrections made for COMP3301 A3.

Introduction

This paper describes Virtual Hard Disk (VHD) image format. It does not explain how hard disks interface with virtual machines, nor does it provide information about ATA (AT Attachment) hard disks or Small Computer System Interface (SCSI) hard disks. This paper focuses on how to store the data in files on the host file system.

The reader should be familiar with virtual machine technology and terminology, such as the terms guest and host as used in the context of virtual machine architectures. The user should also be familiar with hard disk technologies and should understand how data is accessed and laid out on the physical medium. The following terminology is used in this paper:

Absolute Byte Offset

Refers to the byte offset from the beginning of a file.

Reserved

Fields marked reserved are deprecated or are reserved for future use.

Sector length

Sector length is always 512 bytes.

All values in the file format, unless otherwise specified, are stored in network byte order (big endian). Also, unless otherwise specified, all reserved values should be set to zero.

Overview of Virtual Hard Disk Image Types

Virtual hard disks are implemented as files that reside on the native host file system. The following types of virtual hard disk formats are supported by the VHD format:

- Fixed hard disk image

- Dynamic hard disk image
- Differencing hard disk image (not required for COMP3301 A3)

Each virtual hard disk image has its own file format, as described in the following sections.

Fixed Hard Disk Images

A fixed hard disk image is a file that is allocated to the size of the virtual disk. For example, if you create a virtual hard disk that is 2 GB in size, the system will create a host file approximately 2 GB in size.

The space allocated for data is followed by a footer structure. The size of the entire file is the size of the virtual disk plus the size of the footer.

Dynamic Hard Disk Image

A dynamic hard disk image is a file that at any given time is as large as the actual data written to it plus the size of the header and footer. Allocation is done in blocks. As more data is written, the file dynamically increases in size by allocating more blocks. For example, the size of file backing a virtual 2-GB hard disk is initially around 2 MB on the host file system. As data is written to this image, it grows with a maximum size of 2 GB.

Dynamic hard disks store metadata that is used in accessing the user data stored on the hard disk. The maximum size of a dynamic hard disk is 2040 GB. The actual size is restricted by the underlying disk hardware protocol. For example, ATA hard disks have a 127-GB limit.

COMP3301 Note

You will not be tested on virtual disks with size over 127 GB.

The basic format of a dynamic hard disk is shown in the following table.

Dynamic Disk header fields
Copy of hard disk footer (512 bytes)
Dynamic Disk Header (1024 bytes)
BAT (Block Allocation table)
Data Block 1
Data Block 2
...

Data Block n
Hard Disk Footer (512 bytes)

COMP3301 Note

A dynamic disk image may have a different layout. There is no guarantee whatsoever that the dynamic disk header comes right after the mirror of the footer and the BAT comes right after the dynamic disk header. The only thing guaranteed is that the footer sits at the very end of the image as well as at the very beginning. Everything else are referenced by the footer or other data structures referenced by the footer, and therefore should be read based on the "Data Offset" and "Table Offset" fields.

Every time a data block is added, the hard disk footer must be moved to the end of the file. Because the hard disk footer is a crucial part of the hard disk image, the footer is mirrored as a header at the front of the file for purposes of redundancy.

Differencing Hard Disk Image

A differencing hard disk image represents the current state of the virtual hard disk as a set of modified blocks in comparison to a parent image. This type of hard disk image is not independent, it depends on another hard disk image to be fully functional. The parent hard disk image can be any of the mentioned hard disk image types, including another differencing hard disk image.

COMP3301 Note

This feature is not required for the assignment. You must however deny attaching differencing hard disk images if it is not implemented.

Hard Disk Footer Format

All hard disk images share a basic footer format. Each hard disk type extends this format according to its needs.

The format of the hard disk footer is listed in the following table.

Hard disk footer fields	Size (bytes)
Cookie	8
Features	4
File Format Version	4
Data Offset	8
Time Stamp	4
Creator Application	4

Creator Version	4
Creator Host OS	4
Original Size	8
Current Size	8
Disk Geometry	4
Disk Type	4
Checksum	4
Unique Id	16
Saved State	1
Reserved	427

Note: Some old VHD disk images have 511-byte disk footers, so the hard disk footer can exist in the last 511 or 512 bytes of the file that holds the hard disk image.

COMP3301 Note

Footers of disk images used for testing will only be 512 bytes, you do not have to consider 511-byte footers.

Hard Disk Footer Field Descriptions

The following provides detailed definitions of the hard disk footer fields.

Cookie

The disk footer cookie is the string `"conectix"`. The cookie is stored as an eight-character ASCII string with the `"c"` in the first byte, the `"o"` in the second byte, and so on.

Features

This is a bit field used to indicate specific feature support. The following table displays the list of features. Any fields not listed are reserved.

Feature	Value
No features enabled	`0x00000000`
Temporary	`0x00000001`
Reserved	`0x00000002`

No features enabled. The hard disk image has no special features enabled in it.

Temporary. This bit is set if the current disk is a temporary disk. A temporary disk designation indicates to an application that this disk is a candidate for deletion on shutdown.

Reserved. This bit must always be set to 1.

All other bits are also reserved and should be set to 0.

COMP3301 Note

You will only be dealing with disk images with the reserved bit set.

File Format Version

This field is divided into a major/minor version and matches the version of the specification used in creating the file. The most-significant two bytes are for the major version. The least-significant two bytes are the minor version. This must match the file format specification. For the current specification, this field must be initialized to `0x00010000`.

The major version will be incremented only when the file format is modified in such a way that it is no longer compatible with older versions of the file format.

COMP3301 Note

The version must be 0x00010000, otherwise you should treat it as an invalid disk image.

Data Offset

This field holds the absolute byte offset, from the beginning of the file to the next structure. This field is used for dynamic disks and differencing disks, but not fixed disks. For fixed disks, this field should be set to `0xFFFFFFFFFFFFFFFF`.

COMP3301 Note

For fixed disks, it should be set to 0xFFFFFFFFFFFFFFFF, not 0xFFFFFFFF as described in the official specification of the VHD format.

Time Stamp

This field stores the creation time of a hard disk image. This is the number of seconds since January 1, 2000, 12:00:00 AM in UTC/GMT.

Creator Application

This field is used to document which application created the hard disk. The field is a left-justified text field. It uses a single-byte character set.

COMP3301 Note

Creator application of disk images created by vhdtool is always "obsd", which stands for OpenBSD. You should handle VHDs created by all applications (i.e., do not reject VHDs simply because it's created by some unknown application).

Creator Version

This field holds the major/minor version of the application that created the hard disk image.

COMP3301 Note

vhdtool sets this field to 0x00070005. You should not check for the creator version; all values are valid.

Creator Host OS

This field stores the type of host operating system this disk image is created on.

Host OS type	Value
Windows	`0x5769326B` (`Wi2k`)
Macintosh	`0x4D616320` (`Mac `)
OpenBSD	`0x4F425344` (`OBSD`)

COMP3301 Note

You do not have to check for host OS, and you should not reject VHDs with unknown host OSes.

Original Size

This field stores the size of the hard disk in bytes, from the perspective of the virtual machine, at creation time. This field is for informational purposes.

Current Size

This field stores the current size of the hard disk, in bytes, from the perspective of the virtual machine.

This value is same as the original size when the hard disk is created. This value can change depending on whether the hard disk is expanded.

COMP3301 Note

You can always assume the original size is equivalent to the current size, as we will not test expanding disk images.

Disk Geometry

This field stores the cylinder, heads, and sectors per track value for the hard disk.

Disk Geometry field	Size (bytes)
Cylinder	2
Heads	1
Sectors per track/cylinder	1

When a hard disk is configured as an ATA hard disk, the CHS values (that is, **C**ylinder, **H**eads, **S**ectors per track) are used by the ATA controller to determine the size of the disk. When the user creates a hard disk of a certain size, the size of the hard disk image in the virtual machine is smaller than that created by the user. This is because CHS value calculated from the hard disk size is rounded down.

Disk Type

Disk Type field	Value
None	0
Reserved (deprecated)	1
Fixed hard disk	2
Dynamic hard disk	3
Differencing hard disk	4
Reserved (deprecated)	5
Reserved (deprecated)	6

COMP3301 Note

You should only allow attaching disks with type set to fixed or dynamic. You are not required to handle differencing images, however you may if you wish – no additional marks are awarded for doing so and it will not be tested.

Checksum

This field holds a basic checksum of the hard disk footer. It is just a one's complement of the sum of all the bytes in the footer without the checksum field.

The pseudo-code for the algorithm used to determine the checksum can be found in the appendix of this document.

COMP3301 Note

You should deny attaching disks with invalid footer checksums.

Unique ID

Every hard disk has a unique ID stored in the hard disk. This is used to identify the hard disk. This is a 128-bit universally unique identifier (UUID). This field is used to associate a parent hard disk image with its differencing hard disk image(s).

Saved State

This field holds a one-byte flag that describes whether the system is in saved state. If the hard disk is in the saved state, the value is set to 1. Operations such as compaction and expansion cannot be performed on a hard disk in a saved state.

COMP3301 Note

You should deny attaching disks with saved state set to 1 in read and write mode, as doing so may result in the destruction of the state save.

Reserved

This field contains zeroes. It is 427 bytes in size.

Dynamic Disk Header Format

For dynamic and differencing disk images, the "Data Offset" field within the image footer points to a secondary structure that provides additional information about the disk image. The dynamic disk header should appear on a sector (512-byte) boundary.

The format of the Dynamic Disk Header is listed in the following table.

Dynamic Disk Header fields	Size (bytes)
Cookie	8
Data Offset	8
Table Offset	8
Header Version	4
Max Table Entries	4
Block Size	4
Checksum	4

Parent Unique ID	16
Parent Time Stamp	4
Reserved	4
Parent Unicode Name	512
Parent Locator Entry 1	24
Parent Locator Entry 2	24
Parent Locator Entry 3	24
Parent Locator Entry 4	24
Parent Locator Entry 5	24
Parent Locator Entry 6	24
Parent Locator Entry 7	24
Parent Locator Entry 8	24
Reserved	256

Dynamic Disk Header Field Descriptions

The following provides detailed definitions of the dynamic disk header fields.

Cookie

This field holds the value "`cxsparse`". This field identifies the header.

Data Offset

This field contains the absolute byte offset to the next structure in the hard disk image. It is currently unused by existing formats and should be set to `0xFFFFFFFFFFFFFFFF`.

COMP3301 Note

It should be set to 0xFFFFFFFFFFFFFFFF, not 0xFFFFFFFF as described in the official specification of the VHD format.

Table Offset

This field stores the absolute byte offset of the Block Allocation Table (BAT) in the file.

Header Version

This field stores the version of the dynamic disk header. The field is divided into Major/Minor version. The least-significant two bytes represent the minor version, and the most-significant two bytes represent the major version. This must match

with the file format specification. For this specification, this field must be initialized to `0x00010000`.

The major version will be incremented only when the header format is modified in such a way that it is no longer compatible with older versions of the product.

Max Table Entries

This field holds the maximum entries present in the BAT. This should be equal to the number of blocks in the disk (that is, the disk size divided by the block size).

Block Size

A block is a unit of expansion for dynamic and differencing hard disks. It is stored in bytes. This size does not include the size of the block bitmap. It is only the size of the data section of the block. The sectors per block must always be a power of two. The default value is `0x00200000` (indicating a block size of 2 MB).

Checksum

This field holds a basic checksum of the dynamic header. It is a one's complement of the sum of all the bytes in the header without the checksum field.

If the checksum verification fails, the file should be assumed to be corrupt.

Parent Unique ID

This field is used for differencing hard disks. A differencing hard disk stores a 128-bit UUID of the parent hard disk. For more information, see "Creating Differencing Hard Disk Images" later in this paper.

Parent Time Stamp

This field stores the modification time stamp of the parent hard disk. This is the number of seconds since January 1, 2000, 12:00:00 AM in UTC/GMT.

Reserved

This field should be set to zero.

Parent Unicode Name

This field contains a Unicode string (UTF-16) of the parent hard disk filename.

Parent Locator Entries

These entries store an absolute byte offset in the file where the parent locator for a differencing hard disk is stored. This field is used only for differencing disks and should be set to zero for dynamic disks.

The following table describes the fields inside each locator entry.

Parent locator table field	Size (bytes)
Platform Code	4
Platform Data Space	4
Platform Data Length	4
Reserved	4
Platform Data Offset	8

Platform Code. You do not have to worry about this.

Platform Data Space. You do not have to worry about this.

Platform Data Length. You do not have to worry about this.

Reserved. You do not have to worry about this.

Platform Data Offset. You do not have to worry about this.

Reserved

This must be initialized to zeroes.

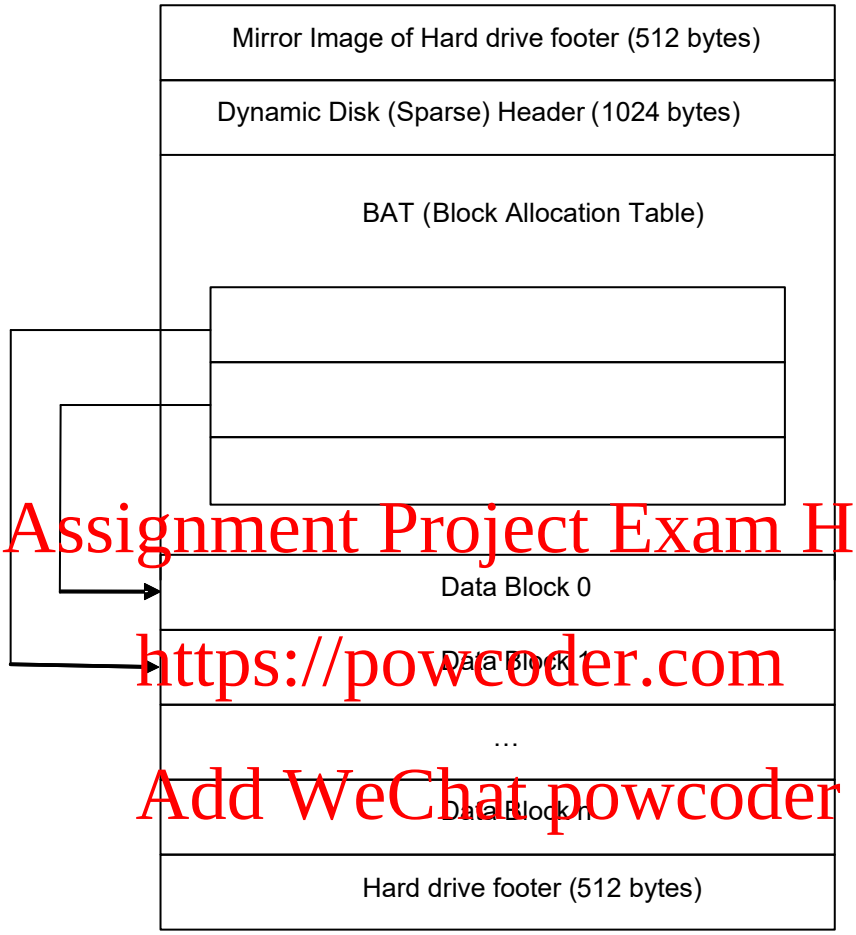
Block Allocation Table and Data Blocks

The Block Allocation Table (BAT) is a table of absolute sector offsets into the file backing the hard disk. It is pointed to by the "Table Offset" field of the Dynamic Disk Header.

The size of the BAT is calculated during creation of the hard disk. The number of entries in the BAT is the number of blocks needed to store the contents of the disk when fully expanded. For example, a 2-GB disk image that uses 2 MB blocks requires 1024 BAT entries. Each entry is four bytes long. All unused table entries are initialized to `0xFFFFFFFF`.

The BAT is always extended to a sector boundary. The "Max Table Entries" field within the Dynamic Disk Header indicates how many entries are valid.

Each entry in the BAT is an unsigned 32-bit integer that points to the absolute sector location of the data block (data block consists of a bitmap and the block data) within the backing file.



COMP3301 Note

A dynamic disk image may have a different layout. There is no guarantee whatsoever that the dynamic disk header comes right after the mirror of the footer and the BAT comes right after the dynamic disk header. The only thing guaranteed is that the footer sits at the very end of the image as well as at the very beginning. Everything else are referenced by the footer or other data structures referenced by the footer, and therefore should be read based on the "Data Offset" and "Table Offset" fields.

The data block consists of a sector bitmap and data. For dynamic disks, the sector bitmap indicates which sectors contain valid data (1's) and which sectors have never been modified (0's).

COMP3301 Note

A data block contains the bitmap and the actual block data, for example, a data block which captures a block of 2 MB has a bitmap of 512 bytes and some block data of 2097152 bytes. The size of the block data is always the block size, and the size of the bitmap is calculated by block size divided by 4096 rounded up to the nearest 512 bytes.

Example: 512 KB (524288 byte) block
- Bitmap size = RoundUp(524288 / 4096, 512) = 512 bytes
- Block data size = 524288 bytes
- Data block size = 512 + 524288 = 524800 bytes

Example: 2 MB (2097152 byte) block
- Bitmap size = RoundUp(2097152 / 4096, 512) = 512 bytes
- Block data size = 2097152 bytes
- Data block size = 512 + 2097152 = 2097664 bytes

Example: 4 MB (4194304 byte) block
- Bitmap size = RoundUp(4194304 / 4096, 512) = 1024 bytes
- Block data size = 4194304 bytes
- Data block size = 1024 + 4194304 = 4195328 bytes

The block bitmap is stored as an array of unsigned 8-bit integers, with the MSB representing the first bit. This means the MSB (bit 7) of the 0th byte is the 0th bit of the bitmap, the LSB (bit 0) of the 0th byte is the 7th bit of the bitmap, the MSB (bit 7) of the 1st byte is the 8th bit of the bitmap, and the LSB (bit 0) of the 1st byte is the 15th bit of the bitmap.

A block is a power-of-two multiple of sectors. By default, the size of a block is 4096 512-byte sectors (2 MB), but blocks can have other sizes. All blocks within a given image must be the same size. This size is specified in the “Block Size” field of the Dynamic Disk Header.

All sectors within a block whose corresponding bits in the bitmap are zero must contain 512 bytes of zero on disk. Software that accesses the disk image may take advantage of this assumption to increase performance.

Implementing a Dynamic Disk

Blocks are allocated on demand. When a dynamic disk is created, no blocks are allocated initially. A newly created image contains only the data structures described earlier (including the Dynamic Disk Header and the BAT).

When data is written to the image, the dynamic disk is expanded to include a new block. The BAT is updated to contain the offset for each new block allocated within the image.

Mapping a Disk Sector to a Sector in the Block

To calculate a block number from a referenced sector number, the following formula is used:

BlockNumber = floor(RawSectorNumber / SectorsPerBlock)

SectorInBlock = RawSectorNumber % SectorsPerBlock

BlockNumber is used as an index into the BAT. The BAT entry contains the absolute sector offset of the beginning of the block's bitmap followed by the block's data. The following formula can be used to calculate the location of the data:

ActualSectorLocation = BAT [BlockNumber] + BlockBitmapSectorCount + SectorInBlock

In this manner, blocks can be allocated in any order while maintaining their sequencing through the BAT.

When a block is allocated, the image footer must be pushed back to the end of the file. The expanded portion of the file should be zeroed.

Appendix: Checksum Calculation

Checksum Calculation

Variables in checksum calculation	Description
`driveFooter`	Variable holding the drive footer structure
`checksum`	Variable that stores the checksum value
`driveFooterSize`	Size of the `driveFooter` structure
`counter`	Local counter

```
checksum = 0;
driveFooter.Checksum = 0;
for (counter = 0; counter < driveFooterSize; counter++)
{
   checksum += driveFooter[counter];
}
driveFooter.Checksum = ~checksum;
```