# Usable File System version 1.0.00 Official Specification.

# Contents

# 1.0 Introduction.

Usable File System or UFS is a simple file system. It is designed to be easily improved without breaking compatibility with older drivers. It is a 64 bit file system designed to be easy to implement. It is made of 2-3 sections: the UDB(UFS Data Block) Area, the SBA(Special Boot Area), and the DS(Data Section).

## 1.1 The VBR(for boot)

The VBR is the first sector of partition, this is fully isolated from the FS. It is only used for the bootloader.

## 1.2 The UDB Area

The UDB Area resides directly after the VBR. The UDB Area is composed of the UDB and the backup UDB (BUDB) (This section is optional, if it is not used it this cluster is cleared to 0, but still used space on disk.).The UDB is basically the SuperBlock in UFS. It points to the Root directory, Free blocks, Bad blocks, and other partition information.

## 1.3 The SBA

The SBA is a reserved spot between the UDB Area and the Data Section. It is reserved for a second-stage boot loader and may not be present.

## 1.4 The DS

The DS is is the third part of disk. It contains all Data (files/lists/directories/file contents/…) for the partition.

## 1.5 Writing to UFS

For the safety of files, when a file is written to the data is to be written to sectors marked as free then the pointers are updated when the blocks are finished being written to and the old blocks are free. This is not mandatory, but it is advised, because this helps prevent corruption of files.

## 1.6 Partition Size

A UFS partition **MUST** be no less than 10KiB. In theory it could be smaller, but it would be practically useless. Do **NOT** make a UFS partition smaller than this, but do support it because it could exist for special purposes.

# 2.0 UFS Partition Identification

There are two things to check when trying to verify a partition is UFS. Those two things are the Partition ID in the Partition Table (MBR/GPT) and the UFS signature. The MBR ID is 0x49 and the GPT Partition Type GUID is 01234567-89AB-CDEF-0000-000000000000. The UFS signature is in the UDB, it is the first eight bytes of the UDB and it reads ‘UFS PART’. If you want more checks you can also verify that the version number, start of the DS, Block/Cluster size, size of partition, and at maximum no mare then one pointer (Root Entry/Free Block List/Bad Block List/Partition Info) is/are not zero.

# 3.0 The UDB

The UDB starts with an ASCII string that reads ‘UFS PART’ for identifying the partition as UFS. It points to  everything else on the partition. The UDB is the first Block of disk and the BUDB is the second. It is only 512 bytes long. NULL padded to Cluster boundary.

The ‘Optional’ UDB Backup (BUDB), is **ALWAYS** present, but if it is not set up it is filled with zeros. It starts with an ASCII string that reads ‘UFSBPART’, but otherwise is an exact copy of the UDB. If the UDB is modified the BUDB also must be updated, but the signature **MUST** be ‘UFSBPART’ instead of ‘UFS PART’.

## UDB structure.

| Offset (HEX) | Size (HEX) | Description |
|---|---|---|
| 0x00 | 0x08 | UFS signature, **MUST** be 'UFS PART' for UDB, or ‘UFSBPART’ for BUDB. |
| 0x08 | 0x08 | Block Size in bytes (BS). minimum = 0x200. Must be a multiple of the disk sector size. |
| 0x10 | 0x08 | Size of partition in Blocks, this includes the UDB and Reserved areas, for Data size use this and subtract start of Data. |
| 0x18 | 0x08 | Flags (If you see Fx, x is the bit number for that flag.). Flags:<br>Bit 0 = Read-Only (RO).<br>All others are Reserved. |
| 0x20 | 0x08 | Size Of Special Boot Area (SBA). |
| 0x28 | 0x08 | Start of Data Section (In Blocks), from partition. |
| 0x30 | 0x08 | Root Entry (RE) Pointer, Offset from Data Section, this is the same as any other entry. |
| 0x38 | 0x08 | Free Blocks List (FBL) Pointer, Offset from Data Section. |
| 0x40 | 0x08 | Bad Blocks List (BBL) Pointer, Offset from Data Section. |
| 0x48 | 0x08 | Partition Info (PI) Pointer (For things like: Mount time, last check, ....), Offset from Data Section. |

| | | |
|---|---|---|
| 0x50 | 0x08 | Version (In HEX) in the form of, if this is 0x1001 the version is 1.0.01, if it is 0x1000 it is 1.0.00, or if 0xA001 is A.0.01, can **NOT** be 0. |
| 0x58 | 0x08 | Oldest version of driver that can read/write/create file/delete file/… |
| 0x60 | 0x10 | Partition name, padded with zeros. |
| 0x68 | 0x1A8 | Reserved for use in future versions. |

# 4.0 Partition Info

The Partition Info (PI) Block contains all the information that is going to be updated often, so that the UDB is not written to very often. It is also only 512 bytes long. NULL padded to Cluster boundary.

## Partition Info

PI:
Offset Size What it is
0x00 0x08 Unsigned 64bit POSIX time of partition birth, 0 if not supported by driver that created it, never written.
0x08 0x08 Unsigned 64bit POSIX time of last mount, set 0 if not supported by driver.
0x10 0x08 Unsigned 64bit POSIX time of last unmount, set 0 if not supported by driver.
0x18 0x08 Unsigned 64bit POSIX time of last check, set 0 if not supported by driver.
0x20 0x1E8 Reserved for future versions.

| Offset (HEX) | Size (HEX) | Description |
|---|---|---|
| 0x00 | 0x08 | Signed 64bit POSIX time of partition birth, 0 if not supported by driver that created it, never written too, even though in PI. |
| 0x08 | 0x08 | Signed 64bit POSIX time of last mount, set 0 if not supported by driver. |
| 0x10 | 0x08 | Signed 64bit POSIX time of last unmount, set 0 if not supported by driver. |
| 0x18 | 0x08 | Signed 64bit POSIX time of last check, set 0 if not supported by driver. |
| 0x20 | 0x08 | Writable Partition Flags. Reserved. |
| 0x28 | 0x08 | Last block of FBL. Offset from DS. |
| 0x30 | 0x08 | Last Used entry of Last block of FBL. From 1 up, 0 means none. |
| 0x38 | 0x08 | Last block of BBL. Offset from DS. |
| 0x40 | 0x08 | Last Used entry of Last block of BBL. From 1 up, 0 means none. |

| | | |
|---|---|---|
| 0x48 | 0x1C0 | Reserved for future versions. |

# 5.0 The FBL and BBL

The Free Block List is a list of all free blocks on the partition. The Bad Block List is a list of all the bad blocks on the partition. There **MUST NOT** be any gaps between entries, when one is cleared either, the entire list beyond that point must be shifted one place to file the gap, or it must be replaced with the one on the end. Extended blocks can not be on block zero of the partitions data section. When expanding list check if there is a next block already reserved with the second field.

## FBL/BBL structure

| Offset (HEX) | Size (HEX) | Description |
|---|---|---|
| 0x00 | 0x08 | Number of Blocks that are Free/Bad on partition. This field is only present on the first block in this list, for all extended blocks this points to the block before it in this list. This is how to tell number of free blocks. |
| 0x08 | 0x08 | Next Block in this list, if last block this is undefined, Must be zero. |
| 0x10 | 0x10 | The first Free/Bad Block Entry. Each one is sixteen bytes long. |
| 0x20 | ? | More Free/Bad Block Entries. |

## 5.1 Free/Bad Block Entry structure

| Offset (HEX) | Size (HEX) | Description |
|---|---|---|
| 0x00 | 0x08 | Block offset from DS. |
| 0x08 | 0x08 | Number of consecutive free/bad blocks. Zero is only one block. |

# 6.0 Directory/Object Entries

All objects (Files/Directories/…, but not Lists/...) are represented with a Directory/Object Entry. For Directories, they point to Objects by giving a list of Blocks with 256 byte entries, Object Names with pointer to its Object Entry.

## Directory/Object Entry Structure

| Offset (HEX) | Size (HEX) | Description |
|---|---|---|
| 0x00 | 0x08 | Flags.<br>Bit 0 = File, if cleared it is a regular directory.<br>Bit 1 = RO.<br>Bit 2 = Hidden.<br>Bit 3 = Exicute. |
| 0x08 | 0x08 | Signed 64bit POSIX time of creation, 0 if not supported by driver. |
| 0x10 | 0x08 | Signed 64bit POSIX time of last write, 0 if not supported by driver. |
| 0x18 | 0x08 | Pointer to Contents List (Block number offset from Data Section), when a file is created it **MUST** be given a contents list even if the list is empty, this field can **NOT** be 0 for empty it will be read as block 0, there **MUST** be a Contents List. |
| 0x20 | 0x08 | Number of objects (files/directories/…) if Directory, if this is a file: size in bytes. |
| 0x28 | ? | Reserved. |

# 7.0 Contents Lists, File Name Blocks, and File Name Block Entries.

## 7.1 Contents Lists

A Contents List is a list of all Objects (Files/Directories/…) in a Directory, or a List of all Blocks belonging to a File. If there is extra room but not enough for another File Name Blocks Entries it is left empty.

### Contents List structure

| Offset (HEX) | Size (HEX) | Description |
|---|---|---|
| 0x00 | 0x08 | Next Block in this list. |
| 0x08 | ? | Entries for directories these are pointers to entry blocks (File Name Blocks), for files and Free/Bad blocks these are pointers to the blocks them selves. If a Directory has no objects, there are no File Name Blocks. 0 means empty. |

## 7.2 File Name Blocks

This is a Block that the contain 256 byte Object entries. To find out how many entries there are per File Name Block all you have to do is: BS/256.

### 7.2 File Name Blocks Entries

| Offset (HEX) | Size (HEX) | Description |
|---|---|---|
| 0x00 | 0xF8 | Object Name padded with zeros. The Last byte **MUST** be NULL, so you only have 0xF7 bytes for the File Name. If all 0 then empty. |
| 0xF8 | 0x08 | Location of Object Entry, that is the offset of its Directory entry from the Data Section. |
| 0x100 | ? | More. |

# 8.0 UFS Terminology

**BBL** – Bad Block List.
**Block** – A block of data that’s size is defined in the UDB.
**BS –** Block Size.
**BUDB** – Backup UDB.
**DS** – Data Section.
**FBL** – Free Block List.
**GPT Partition Type GUID** – 01234567-89AB-CDEF-0000-000000000000.
**List** – Array of pointers.
**MBR Partition Type ID –** 0x49.
**Object** – Anything that is in a directory, e.g. Files, Directories, etc..
**PI** – Partition Info block, contains advanced partition info.
**RE, Root Entry** – The first/base directory, it contains all objects.
**RO** – Read-Only.
**SBA** – Special Boot Area
**UDB** – UFS Data Block.
**UFS** – Usable File-System.