Document Number: D-1632
Creation Date: 23 Tellurium 80
Last Revision: 24 Radon 1000
Revises: N456
Reply-to: anon@dome.gl

Working Draft - Coalition File System

Note: This is a draft, known to be incomplet and incorrekt, and has lots of bad formatting.

1- General

1.1- Scope

This draft discusses the implementation of the CFS file system, drawn from information found in the archives of the dome.
All information contained within this document is subject to interpretation by eventual readers and should be treated with particular attention.

CFS is a general-purpose file system, designed to be used on storage mediums of any kind, and describes how data can be stored and organised on said mediums.

In addition to storing and organising information on said mediums, facilities designed to improve data security and/or density are defined herein.

1.2- Normative references

The following referenced documents are indispensable for the application of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

- ISO/IEC 10646: Universal Coded Character Set (UCS)
- ISO/IEC 2382 (all parts), Information technology — Vocabulary
- ISO/IEC 9899 - Programming languages - C

1.3- Terms and Definitions

For the purposes of this document, the following definitions apply.

1.3.1 Byte

A string of eight binary digits operated upon as a unit.

1.3.2 Descriptor

A structure containing descriptive information about a volume or a file.

1.3.3 File

A named collection of information.

1.3.4 Directory

A named collection of files.

1.3.5 Sector

The smallest addressable part in the underlying medium that can be accessed independently.
Note: All the mediums uncovered as of now have a sector size of 512 bytes.

1.3.6 Extent

An extent is defined in this document as a contiguous array of sectors.

1.3.7 Endianness

The order in which bytes have to be read, there are three previoulsy recorded entries:

- Little-Endianness: The least significant byte is in the rightmost position of the entity being read, all following bytes are most-significant than the previous one. This will be referred to as LE in the remaining part of this document.
- Big-Endianness: The least significant byte is in the leftmost position of the entity being read, all following bytes are least-significant than the previous one. This will be referred to as BE in the remaining part of this document.
- Middle-Endianness: This type of endianness will be used if specified for power-of-two long entities. It splits the value into two halves, each in LE order. This will be referred to as ME in the remaining part of this document.
    - 32-bit example: 0x12345678 becomes two 16-bit packets: 0x3412 0x7856.
    - 64-bit example: 0x123456789ABCDEF0 becomes two 32-bit packets: 0x78563412 0xF0DEBC9A.

1.4- Notation

The following notation is used in this document.

1.4.1- Bases

Magic numbers and masks will be defined in different bases throughout this document, these are the conventions for defining such numbers:

- Decimal: default reading for numbers in this document, digits individually range from ‘0’ through ‘9’.
- Hexadecimal: in this document, all uses of hexadecimal will be prefixed by the string “0x” for clarity purposes. All hexadecimal digits range from ‘0’ through ‘9’ and from ‘A’ through ‘F’.
- Octal: all uses of octal in this document shall use the C convention of naming as defined in ISO/IEC 9899.
- Binary: binary uses in this document shall use a prefix in the shape of “0b”. All binary digits range from ‘0’ through ‘1’.

1.4.2- Signedness

All numbers described in this document can be described as values of a particular length and signedness, as defined here:

1.4.2.1- Unsigned integer

An unsigned integer ranges from a minimal value of 0 to a maximum value of 2^(n - 1)-1 where n is the number of bits of the unsigned entity.

1.4.2.2- Signed integer

An unsigned integer ranges from a minimal value of -(2^(n-2)) to a maximum value of 2^(n - 2)-1 where n is the number of bits of the signed entity.

1.4.3- Date format

In CFS, dates are recorded as a 14 byte-long field, split as follows:

- Magic string “DATETM”, 6 bytes in ASCII

- Year of creation in Alpha calendar form, i.e. years passed from the great disaster, 2 byte BE recorded according to 1.4.2.1.
- Month of creation, in Alpha calendar form, i.e. from 1 through 13, 1 byte recorded according to 1.4.2.1.
- Day of creation, in Alpha calendar form, i.e. from 1 through 30, 1 byte recorded according to 1.4.2.1.
- Seconds in the day of creation, in Alpha calendar, from 1 through 86400, 4 bytes BE recorded according to 1.4.2.1.

1.5- Identifier rules

1.5.1- Files identifiers

A file identifier can be composed of any UTF-8 character except:

- The character SEPARATOR-1 identified by the bit combination 0x60
- The character SEPARATOR-2 identified by the bit combination 0x2F

1.5.2- Directory identifiers

A directory identifier can be composed of any UTF-8 character except:

- The character SEPARATOR-1 identified by the bit combination 0x60
- The character SEPARATOR-2 identified by the bit combination 0x2F

2- Volume headers

2.1- Metadata

For ease of use, the first extent of the volume contains metadata.
Such metadata keeps a record of the actual use of the available data on the volume.

2.1.1- Volume descriptor

The volume descriptor is necessarily defined in the first extent.

2.1.1.1- Size of the volume

Size of the volume, in number of extents, stored in ME form, recorded with 8 bytes according to 1.4.2.1.

2.1.1.2- Extent size

1 byte describing the size of an extent, in number of sectors, recorded according to 1.4.2.1.

2.1.1.3- Volume name

40 bytes reserved for the storage of a UTF-8 string.

2.1.1.4- File hierarchy size

8 bytes in BE form describing the size in extents defining the file hierarchy of the system as defined in 2.1.2.

2.1.1.5- Data encryption

1 byte describing the security measure type.

Values are either:

- 0x8x: ROLx
- 0x40: XOR
- 0x00: None

2.1.1.6- Mask data

In case the encryption scheme chosen is XOR, the mask is defined in the following section
The first byte defines the length of the mask, and the next bytes are the mask itself.

2.1.2- File Hierarchy

File metadata is stored in the file hierarchy section of the volume. Its size is defined in 2.1.1.4.

2.1.2.1- Directory metadata

Directories metadata are described here.

2.1.2.1.1- Directory header

A directory is defined as an entity whose first byte has value 0x80.

2.1.2.1.2- Directory identifier

The following 4 bytes BE is reserved for a unique identifier for the directory, recorded according to 1.4.2.1.

2.1.2.1.3- Parent directory

The identifier of a directory is to be stored here, and shall have the value of a directory identifier as defined in 2.1.2.1.2, its value must be non-zero for non-root directories.

2.1.2.1.4- Directory name

The name of a directory is limited to a total of 50 bytes formatted in UTF-8, null terminated (hence, 49 usable bytes for storing information).

2.1.2.1.5- Data location

Offset from the beginning of the data section, expressed in extents. It is a 8 bytes long field in ME form, recorded according to 1.4.2.1.

2.1.2.2- Root directory

A file hierarchy has one root directory, containing all the files and subsequent directories.

It is defined first in the file hierarchy, and can never be removed.

Its data is the first extent following the file hierarchy description and its parent directory information must be 0x00.

2.1.2.3- File metadata

Meta-information about files is defined here.

2.1.2.3.1- File header

A file is defined as an entity whose first byte has value 0x40.

2.1.2.3.2- File id

The identifier of a file is to be stored here, and shall be formatted similarly to 2.1.2.1.2, its value must be non-zero.

2.1.2.3.3- Parent directory

The identifier of a directory is to be stored here, and shall have the value of a directory identifier as defined in 2.1.2.1.2, its value must be non-zero.

2.1.2.3.3- File name

The name of a file is limited to a total of 50 bytes in UTF-8 format, null terminated (hence, 49 usable bytes for storing information).

2.1.2.3.4- Data location

The data location contains the information of a file’s location in the data section of the volume. It is a 8 bytes long field in ME form, recorded according to 1.4.2.1.

2.1.2.3.5- File size in extents

The file total size in extents is defined here as a 8 bytes long field in ME form, recorded according to 1.4.2.1.

2.1.2.3.6- File size in bytes

The file total size in bytes is defined here as a 8 bytes long field in ME form, recorded according to 1.4.2.1.

3- Volume Data

3.1- Directory information

3.1.1 Directory definition information

A directory's information is data about the files and directories it contains, and is defined in this section.

3.1.1.1- Directory continuation header

If the current extent is a continuation header, it should bear the magic number 0x80 as its first byte, else it will contain 0x40

3.1.1.2- Number of children

The number of children of a directory is contained as a 4-bytes BE number recorded according to 1.4.2.1.

3.1.1.3- Directory creation date

The creation date of a directory is stored in the standard date format as defined in 1.4.3.

3.1.1.4- Directory children

The children of the repository are enumerated after the last modification date.
They will be enumerated through a collection of records in the following form

- Type: any number as defined in either 2.1.2.1.1 or 2.1.2.3.1
- Identifier of the entity if it is a directory, the location of the metadata if it is a file

3.1.1.5- Continuation location

If a directory contains many files, the last 4 bytes BE should contain the position of the next extent with information relative to this directory; 0 if no continuatioon is needed.

3.1.2 Directory continuation information

3.1.2.1- Directory continuation header

If the current extent is a continuation header, it should bear the magic number 0x80 as its first byte, else it will contain 0x40

3.1.2.2- Directory children

The children of the repository are enumerated through a collection of records in the following form:

- Type: any number as defined in either 2.1.2.2.1 or 2.1.2.3.1
- Identifier of the entity, in 4 bytes ME

3.1.2.3- Continuation location

The last 4 bytes BE should contain the position of the next extent with information relative to this directory, or 0x00 if none is needed.

3.2- File information

3.2.1 File data

3.2.2.1 File continuation header

If an extent is a file continuation header, it should bear the magic number 0x20 as its first byte, 0x00 otherwise.

3.2.2.1- Local size information

The size of the current extent that is in use by the file, in bytes. 4 bytes long in ME form.

3.2.2.2- File data

The binary data of the file, whose size is defined in 2.1.2.3.6

3.2.2.3- Continuation information

The last 4 bytes in BE form are reserved for the continuation information of the file data, if a file cannot fit in one extent, its data will be stored in another extent, whose information should be stored here. It should be set to 0x00 if no data if no continuation sector is available.

4- Data Security Measures

This section defines the different security measures available for securing data in the volume, using either protection scheme defined in 2.1.1.6.

4.1 ROLx

ROLx is a left rotation of the bits of each by x.
The operation shall be repeated on each byte, independently before reading the contents of the volume.
The x is defined as the 3 last bits of the Data Encryption byte (see 2.1.1.6)

4.2 XOR

A XOR a binary operation that shall be done on the content of the volume.
Each byte needs to be XOR'd with the appropriate byte from the given mask to find its original value.

Appendix A - Oversight

This appendix provides a graphical overview of the global idea of CFS, all details will be handled within the specification itself and shall be discussed in the core sections of the document.

Extent 1- Volume metadata

The first extent provides the File System with informations about itself, namely:

- Its size
- The size of the metadata
- The size and position of the data

Extent 2->N- File hierarchy

The following n (n should be interpreted as the value defined in section 2 of the present document) extents are metadata about the files and their hierarchy.

Directories and files are defined here, every metadata is stored in this section.

All data is contiguous and can be replaced easily as all is homogenoeus in terms of size.

Extent N+1 -> M- Data

All the data of the media is stored in this section, extents are separated by headers and tails, as defined in section 3.
Every extent will contain at most one file or directory, as specified in section 3.