DoubleSpace Compressed Volume File Overview

Version 1.00.01

April 30, 1993

Contents

1 Introduction

This document provides an overview of the layout of the MS-DOS 6 DoubleSpace Compressed Volume File (CVF) format. For details on the CVF format, please see the source code for the DSDUMP.EXE program (especially MAIN.C). DSDUMP.EXE is a utility program that allows you to view regions of a CVF either in straight hex code, or in more useful formatted displays, suitable to the particular region you are viewing.

A CVF contains all of the normal information of a FAT drive (boot sector, FAT, root directory), plus data structures that allow DoubleSpace to store a compressed FAT cluster in less space than occupied on a FAT drive. This *space management* is at the core of how DoubleSpace saves space over a FAT drive. DoubleSpace allocates space in a CVF in units of 1 sector (512 bytes), as compared with FAT, which allocates space in units of 1 cluster.

2 CVF File Names

A DoubleSpace CVF has a name of the form DBLSPACE.nnn, where nnn is a number in the range 000-254. This number is called the *sequence number* of the CVF. The sequence number 000 is special: it indicates that the CVF was created by compressing the contents of an existing drive. When DoubleSpace mounts a DBLSPACE.000 CVF, the drive letter of the *host drive* is used to refer to the CVF, and a new drive letter is used to refer to the host drive.

DoubleSpace can only mount CVFs that exist in the root directory of a FAT drive.

3 Compressed Volume File (CVF) Layout

A DoubleSpace CVF has the following regions, which occur in the CVF in the order listed below. For details on the format and usage of these regions, please see the source code for DSDUMP.EXE.

Name	Offset (sector)	Size (sectors)	Description
MDBPB	**0**	**1**	A structure which has an MS-DOS BPB plus DoubleSpace-specific information. In particular, the MaximumCapacity of the CVF is stored here, which is used to determine the size of subsequent regions.
BitFAT	**1**	**varies**	The BitFAT has a bit for each sector in the Sector Heap (see below). A bit is 1 if the corresponding sector is in use, and is 0 if the corresponding sector is free. The space allocated for the BitFAT is a function of the MaximumCapacity of the CVF. For the largest MaxCap (512Mb), the BitFAT will be occupy 128Kb. The actual space in use will vary, depending on the current size of the sector heap.
Reserved1	**...**	**1**	Reserved.
MDFAT		**varies**	A table of 4 byte entries which maps FAT clusters to sectors in the Sector Heap which contain the (usually compressed) data for a cluster. There is an entry in this table for each cluster on the compressed drive. The size of this table is a function of the MaximumCapacity of the CVF. For the largest MaxCap (512Mb), the MDFAT will occupy 256Kb.
Reserved2	**...**	**31**	Reserved.

Boot Sector	**...**	**1**	The boot sector for the DoubleSpace drive. This is not used for booting, even if the DoubleSpace drive is mounted as drive C. When MS-DOS reads sector 0 from the compressed volume, this is the sector that is returned. This structure is described in the MS-DOS version 5 Programmer's reference on page 34.
Reserved3	**...**	**varies**	Reserved. This exists to position the FAT at a desired sector-multiple offset in the CVF. First DoubleSpace stamp (F8,'D','R',0) is at start of the first sector of this region.
FAT	**...**	**varies**	The FAT for the DoubleSpace drive. This is a standard MS-DOS FAT.
Root Directory	**...**	**32**	The root directory for the DoubleSpace drive.
Reserved4	**...**	**2**	Reserved.
Sector Heap	**...**	**60**	The sectors which are used to store (usually compressed) data for clusters. The sectors for a cluster are stored contiguously in the heap. When this space is all used, there is no more free space on the partition (even though MS-DOS may think there is because there are clusters free in the FAT).
2nd Stamp	**...**	**1+**	Second DoubleSpace stamp ('M','D','R',0)

4 CVF Region Notes

The following sections provide a notes on the CVF regions. Specific details on each region are addressed in the DSDUMP.EXE source code (MAIN.C).

5 The MDBPB

The MDBPB contains fields that describe the rest of the CVF. See the source file CVF.H for details, and see the usage in MAIN.C. A particularly important field is cmbCVFMax, the *MaximumCapacity* of the CVF. This is the maximum size of the CVF when mounted as a DoubleSpace drive, i.e., then maximum amount of uncompressed data that can be stored in the CVF. This is **not** a direct limit on the size of the CVF itself, which instead depends upon the host drive size, and on the actual compression ratio of the data stored in the CVF.

The MaximumCapacity of a CVF is set at the time the CVF is created by DBLSPACE.EXE, and is based on the size of the host drive. This keeps the *system overhead* of the CVF in scale with the size of the host drive. In particular, the BitFAT, MDFAT, and FAT are all allocated to support this maximum capacity, even though in most cases the CVF is not large enough to require the use of the complete BitFAT, MDFAT, and FAT. The *active* portion of the BitFAT, MDFAT, and FAT is generally much smaller than the allocated size. This preallocation is done to make growing (and shrinking) the CVF very fast, as all that needs to be done is change the size of the CVF (to adjust the Sector Heap size), and change a few MDBPB fields.[1]

1 One tricky part of changing the size of a DoubleSpace drive involves what happens when the number of clusters on the drive crosses the 4086 boundary between 12-bit and 16-bit FAT entries. DoubleSpace has to then convert the FAT between these two formats. When this happens, DBLSPACE.EXE goes to some lengths to ensure that

6 The BitFAT

The BitFAT tracks which sectors in the Sector Heap are in use. Each *word* of the BitFAT maps 16 sectors in the Sector heap. The high bit (bit 15) of the word corresponds to the first sector of the 16 sectors, bit 14 corresponds to the second sector, ..., and bit 0 corresponds to the 16th sector.

The BitFAT is rebuilt (by scanning the MDFAT) each time the CVF is mounted, to ensure that it is correct.

7 Reserved1

This region is reserved for future use by DoubleSpace.

8 The MDFAT

The MDFAT has one 4 byte entry for each FAT cluster. This entry points to the location in the Sector Heap where the data for the cluster is stored, and has flags to indicate whether the data is compressed and if the cluster is in use, and has the compressed and uncompressed length of the cluster.

The following table details the bit fields of each 32-bit MDFAT entry:

Bit Offset	Size (bits)	Name	Description
0	21	secStart	22 bit "sector number" of the first sector in the Sector Heap used to store the data for this sector. All sectors for a cluster are contiguous. Add one to this number to get the CVF sector number.
21	1	Reserved	Reserved for future use.
22	4	csecCoded	4 bit count of the number of sectors (minus 1) used to store the information for this cluster. If csecPlain is less than 15, it is possible for csecCoded to be larger than csecPlain.
26	4	csecPlain	4 bit count of the number of sectors required to store the data for this cluster in uncompressed format.
30	1	fUncoded	Flag indicating if data for this cluster is stored compressed (0) or uncompressed (1). Data is stored uncompressed when DoubleSpace cannot compress it, to avoid taking more space than the uncompressed data requires.
31	1	fUsed	Flag indicating that this MDFAT entry is in use (1) or unused (0). This bit is used by DBLSPACE.BIN in part to support undelete operations.

9 Reserved2

This region is reserved for future use by DoubleSpace.

10 Boot Sector

Standard MS-DOS boot sector. This is present for compatibility, but is not used for booting the system, even if the C drive is compressed.

---

the operation is restartable, so that if the computer is restarted during this conversion, DBLSPACE.EXE will regain control and complete the operation without losing any data.

11 Reserved3

This region is reserved for future use by DoubleSpace. The first 4 bytes of this region contain the first DoubleSpace signature.

12 FAT

This is a normal MS-DOS File Allocation Table. DoubleSpace drives only contain a single FAT in the CVF, but DBLSPACE.BIN virtualizes this as two FATs, to be compatible with software which does not expect a hard drive to have a single FAT. Writes to the virtual 2nd FAT are ignored, and reads from the virtual 2nd FAT are mapped to reads from the true 1st FAT.

13 Root Directory

Standard MS-DOS root directory (512 entries of 32 bytes each).

14 Reserved4

This region is reserved for future use by DoubleSpace.

15 Sector Heap

Heap of sectors which are used to store data for the cluster. Each time DoubleSpace needs disk space to store data for a cluster on the compressed volume, it gets a free sector from the sector heap and sets the corresponding bit in the BitFAT to indicate the sector is in use.

Each compressed cluster begins with a four byte tag which indicates how the data on the sector is compressed:

44 53 00 00	IHVLevel2
44 53 00 01	IHVLevelMax
44 53 00 02	StandardCompression
44 53 00 03	IHVLevel3
44 53 00 04	IHVLevel4

Uncompressed clusters have no leading tag field, but it is of course possible that an uncompressed data cluster could contain what appears to be a valid compression tag. To determine whether the data for a cluster is stored compressed or not, you must examine the MDFAT.

16 2nd Stamp

This regions starts at the last full sector of the CVF, and contains the second DoubleSpace signature.

17 DBLSPACE.BIN Handling of FAT Sector Writes

DoubleSpace scans a FAT sector when MS-DOS writes it, comparing it against the MDFAT. If a FAT entry is marked free, then DoubleSpace looks at the corresponding MDFAT entry, marks it unused, and frees the sectors for the cluster by zeroing the bits in the BitFAT. The rest of the MDFAT entry is left intact.

To undelete a file on an DoubleSpace drive, DBLSPACE.BIN checks the MDFAT entry to determine which sectors were used to store the compressed data. If all of the sectors for the MDFAT cluster contain data, but are marked as free in the BitFAT, DBLSPACE.BIN assumes it can recover the cluster. Otherwise, undelete on a DoubleSpace drive works the same as on a normal FAT partition[1].

1 **How Undelete works on a FAT partition.** When MS-DOS deletes a file, it marks the clusters free in the FAT, but does not touch the data on the clusters. Undelete code assumes the file was allocated contiguously beginning with the first cluster (which is still in the directory entry) for n = filesize/clustersize + 1 clusters. If all of the clusters are still free, undelete assumes they belong to the deleted file. If any cluster has been used and then freed since the file was deleted, or if the file was not allocated contiguously, the file will not be undeleted correctly.