OSTA

CD UDF with ISO-9660

File System for Recording on CD-R and CD-E media

Version 1.90

September 4, 1996

Table of Contents

List of Tables

List of Figures

1. File System Overview

This proposal is for a file system that facilitates use of a CD recorder (either CD-R or CD-E) as a "logical device" on a computer system. A logical device is one that is used in the same manner as a magnetic disk.

This proposal is based on OSTA UDF (Universal Disk Format) and ISO 13346. OSTA UDF is a named domain of ISO 13346, and is thus entirely a subset of ISO 13346. Please see OSTA UDF for more information.

The implementation will use UDF as the interactive file system with an allowance for writing ISO 9660 structures when interchange is desired.

References enclosed in [ ] are references to ISO 13346. The references are in the form [x/a.b.c], where x is the section number and a.b.c is the paragraph or figure number.

1.1 Advantages

OSTA UDF has been adopted as the file system for DVD players. Using this format instead of proprietary intermediate file systems will reduce confusion in the industry and provide a clear migration path to DVD.

UDF is the end result of several years of collaborative effort, including the effort put into developing ISO 13346, which in turn was based on ECMA 167. This proposal is a small incremental change to an existing standard. The added structures conform to the standard's conventions.

The CD-R file system is almost completely underrun proof; the CD-E file system is completely underrun proof. The only parts that must be recorded without an underrun are the two volume descriptor sequences. As these structures are no more than 16 sectors each, this should not present a problem. Recording the second sequence may be retried if needed. In any case, no user data has been recorded to the disc before this operation must occur. The cost of an underrun is simply the link blocks (7); no data written to the disc needs to be re-recorded. No structures have pointers to unrecorded data which could become invalid if an underrun occurs.

No reserved tracks are required (though are allowed for certain performance enhancements), which eliminates the need for guessing sizes required for the file system relative to the data. It provides a deterministic method for finding all information; no searching for the most recent descriptors is necessary.

2. References

ISO 9660:1988	Information Processing - Volume and File Structure of CD-ROM for Information Interchange
IEC 908:1987	Compact disc digital audio system
ISO/IEC 10149:1993	Information technology - Data Interchange on read-only 120mm optical data disks (CD-ROM based on the Philips/Sony "Yellow Book")
Orange Book part-II	Recordable Compact Disc System Part-II, N.V. Philips and Sony Corporation
Orange Book part-III	Recordable Compact Disc System Part-III, N.V. Philips and Sony Corporation
ISO/IEC 13346:1995	Volume and file structure of write-once and rewritable media using non-sequential recording for information interchange. References enclosed in [ ] in this document are references to ISO 13346. The references are in the form [x/a.b.c], where x is the section number and a.b.c is the paragraph or figure number.
OSTA UDF	OSTA Universal Disk Format version 1.01. OSTA UDF is also called UDF in this document.

3. Definitions

Audio session	Audio session contains one or more audio tracks, and no data track.
Audio track	Audio tracks are tracks that are designated to contain audio sectors specified in the ISO/IEC 908.
CD-E	CD-Erasable. An overwritable CD defined in Orange Book, part-III.
CD-R	CD-Recordable. A write once CD defined in Orange Book, part-II.
CD UDF	The file system described by this document. Is a file system based on OSTA UDF and defines extensions for CD-R and CD-E.
Clean File System	The file system on the media conforms to this standard.
Data track	Data tracks are tracks that are designated to contain data sectors specified in the ISO/IEC 10149.
Dirty File System	A file system that is not a clean file system.
Fixed Packet	An incremental recording method in which all packets in a given track are of a length specified in the Track Descriptor Block. Addresses presented to a CD drive are translated according to the Method 2 addressing specified in Orange Book parts-II and -III.
ICB	A control node in ISO 13346. See [4/14.6], [4/14.7], [4/14.8], and [4/14.9]
Logical Block Address	An address relative to the beginning of a partition, as defined in ISO 13346.
Media Block Address	The address of a sector as it appears on the medium, before any mapping performed by the device.
Physical Address	An address used when accessing the medium, as it would appear at the interface to the device.
Random Access File System	A file system for randomly writable media, either write once or rewritable
Sequential File System	A file system for sequentially written media (e.g. CD-R)
Session	The tracks of a volume shall be organized into one or more sessions as specified by the Orange Book part-II. A session shall be a sequence of one or more tracks, the track numbers of which form a contiguous ascending sequence.
Track	The sectors of a volume shall be organized into one or more tracks. A track shall be a sequence of sectors, the sector numbers of which form a contiguous ascending sequence. No sector shall belong to more than one track. *Note: There may be gaps between tracks; that is, the last sector of a track need not be adjacent to the first sector of the next track.*
UDF	OSTA Universal Disk Format 1.01

Variable Packet	An incremental recording method in which each packet in a given track is of a host determined length. Addresses presented to a CD drive are as specified in Method 1 addressing in Orange Book parts II and III.
VAT ICB	A File Entry ICB that describes a file containing a Virtual Allocation Table.
Virtual Address	An address described by a Virtual Allocation Table entry.
VAT	The Virtual Allocation Table (VAT) provides a Logical Block Address for each Virtual Address. The Virtual Allocation Table is used with sequential write once media.

4. Terms

May	Indicates an action or feature that is optional.
Optional	Describes a feature that may or may not be implemented. If implemented, the feature shall be implemented as described.
Shall	Indicates an action or feature that is mandatory and must be implemented to claim compliance to this standard.
Should	Indicates an action or feature that is optional, but its implementation is strongly recommended.
Reserved	A reserved field is reserved for future use and shall be set to zero. A reserved value is reserved for future use and shall not be used.

5. Description

5.1 General

Two file system strategies are defined in this document. The first is a sequential file system model. This model is designed for devices that are not randomly writable; these devices require that, in general, data be appended to previously written data. The second strategy is defined for rewritable, randomly addressable media and is the random access file system model. The unit of writing may be larger than a sector. The random access file system model also contains a description of host based defect management.

CD-R shall use the sequential file system model. CD-E should use the random access file system model, but may use the sequential file system model. The two strategies shall not be used within one medium.

ISO 13346 requires an Anchor Volume Descriptor Pointer (AVDP [3/10.2]) at sector 256 and either n or (n - 256), where n is the last recorded Physical Address on the media. This standard requires that the AVDP be recorded at both sector 256 and sector (n - 256) when each session is closed. A random access file system session is always closed; for the sequential access file system, the file system may be in an intermediate state; see 5.3.3 CD UDF end of session data.

Recording shall be performed in compliance with UDF and ISO 13346. The multisession rules below shall apply for finding the volume recognition sequence and the Anchor Volume Descriptor Pointer.

ISO 9660 requires a Primary Volume Descriptor (PVD) at sector 16. If an ISO 9660 file system is desired, it may contain references to the same files as those referenced by ISO 13346 structures, or reference a different set of files, or a combination of the two.

It is assumed that early implementations will record some ISO 9660 structures but that as implementations of CD UDF become available, the need for ISO 9660 structures will decrease.

Writing shall use mode 2 form 1 data. Mode 2 is specified for CD ROM multisession and has the widest range of compatibility with existing CD ROM drives.

5.2 Information Control Block

One of the fundamental structures of ISO 13346 and UDF is the ICB (Information Control Block) [4/8.10, 4/14.6, 4/14.9]. All files and directories are described by an ICB. The ICB contains file attributes, permissions, association with an extended attribute, and the file location(s). A file can be identified by either a list of extents that contain the file or the file's data may be included in the ICB directly. To include a file in the ICB, the sum of the file size and the ICB size must be less than one sector. The ICB contains a pointer to its "parent" or old ICB. This provides a mechanism for rolling the file system back to an earlier state.

ICBs are recorded with various strategies. Most of these strategies are designed for random access, write once applications. As CD-R drives are not randomly accessible for write, these strategies do not work well. Strategy type 4 shall be used for CD UDF (both sequential and random access file system models). The ICB contains no indirections to ICBs to come, and is recorded as a single sector. Strategies are defined in ISO 13346, [4/14.6.2]. All indirections on CD-R media are handled by the VAT (see section 5.3.1).

5.3 Sequential File System Specific

CD UDF allows an intermediate state on CD-R media in which only one AVDP is recorded; this single AVDP shall be at sector 256 or sector 512 according to the multisession rules below.

Sequential file system writing shall be performed with variable packet writing. This allows maximum space efficiency for large and small updates. Variable packet writing is more compatible with CD-ROM drives as current models do not support method 2 addressing required by fixed packets.

The Logical Volume Integrity descriptor [3/10.10] shall be recorded and the volume marked as open. Logical volume integrity can be verified by finding the VAT ICB at the last recorded Physical Address. If the VAT ICB is present, the volume is clean; otherwise it is dirty.

The Partition Header descriptor [4/14.3], if recorded, shall specify no Unallocated Space Table [4/10.1], no Unallocated Space Bitmap [4/10.1], no Partition Integrity Table [4/11], no Freed Space Table [4/10.1], and no Freed Space Bitmap [4/10.1]. The drive is capable of reporting free space directly, eliminating the need for a separate descriptor.

ISO 13346 as specified requires a randomly addressable media, even for write once media. As this file system is written sequentially (within a track), some modification is needed for efficient application of ISO 13346.

This standard introduces no new ISO 13346 structures. The added information, which fits within ISO 13346 standard structures, describes:

1. a virtual partition or partitions
2. the contents of a special file, called the Virtual Allocation Table

The Virtual Partition does not have an immediate physical mapping. It does not have a pre-established location or extent; therefore, no Partition Descriptor exists for that partition. It simply indicates that the virtual partition exists and which physical partition contains the VAT (see 5.3.1).

Each surface shall contain 0 or 1 read only partitions, 0 or 1 write once partitions, and 0 or 1 virtual partitions. CD media should contain 1 write once partition and 1 virtual partition.

5.3.1 Virtual Allocation Table

The Virtual Allocation Table (VAT) is a map that translates Virtual Addresses to logical. It shall be recorded as a file identified by a File Entry ICB (VAT ICB) [4/14.9] which allows great flexibility in building the table. The VAT ICB is the last sector recorded in any transaction, which is easy to find on any CD-R mechanism. The VAT shall be used on CD-R media. The VAT itself may be recorded at any location.

Each file and directory shall be described by a single direct ICB. The ICB shall be written after the file data to allow for data underruns during writing, which will cause logical gaps in the file data. The ICB can be written afterward which will correctly identify all extents of the file data. The ICB shall be written in the data track, the file system track (if it exists), or both.

The VAT shall be identified by a File Entry ICB with a file type of 0. This ICB shall be the last valid data sector recorded. Error recovery schemes can find the last valid VAT by finding ICBs with file type 0 and examining the contents for the regid at the end of the table.

This file, when small, can be embedded in the ICB that describes it. If it is larger, it can be recorded in a sector or sectors preceding the ICB. The sectors do not have to be contiguous, which allows writing only new parts of the table if desired. This allows small incremental updates, even on discs with many directories. Each sector can hold entries that represent up to 512 directories.

When the VAT is small (a small number of directories on the disc), the VAT is updated by writing a new file ICB with the VAT embedded. When the VAT becomes too large to fit in the ICB, writing a single sector with the VAT and a second sector with the ICB is required. Beyond this point, more than one sector is required for the VAT. However, as multiple extents are supported, updating the VAT may consist of writing only the sector or sectors that need updating and writing the ICB with pointers to all of the pieces of the VAT.

The Virtual Allocation Table is used to redirect requests for certain information to the proper logical location. The indirection provided by this table provides the appearance of direct overwrite capability. For example, the sector describing the root directory could be referenced as virtual sector 1. Over the course of updating the disc, the root directory may change. When it changes, a new sector describing the root directory is written, and its Logical Block Address is recorded as the Logical Block Address corresponding to virtual sector 1. Nothing that references virtual sector 1 needs to change, as it still points to the most current virtual sector 1 that exists, even though it exists at a new Logical Block Address.

The use of virtual addressing allows any desired structure to become effectively rewritable. The structure is rewritable when every pointer that references it does so only by its Virtual Address. When a replacement structure is written, the virtual reference does not need to change. The proper entry in the VAT is changed to reflect the new Logical Block Address of the corresponding Virtual Address and all virtual references then point to the new structure. All structures that require updating, such as directory ICBs, shall be referenced by a Virtual Address. As each structure is updated, its corresponding entry in the VAT ICB shall be updated.

The VAT shall be recorded as a sequence of Uint32 entries in a file. Each entry shall be the offset, in sectors, into the physical partition in which the VAT is located. The first entry shall be for the virtual partition sector 0, the second entry for virtual partition sector 1, etc. The Uint32 entries shall be followed by a regid and a Uint32 entry indicating the location of the previous VAT ICB. The regid shall contain:

- Flags = 0
- Identifier = *UDF Virtual Alloc Tbl (byte sequence 0x2a 0x55 0x44 0x46 0x20 0x56 0x69 0x72 0x74 0x75 0x61 0x6c 0x20 0x41 0x6c 0x6c 0x6f 0x63 0x20 0x54 0x62 0x6c)
- IdentifierSuffix is recorded as in UDF 2.1.4.3

The entry for the previous VAT ICB allows for viewing the file system as it appeared in an earlier state. If this field is 0xffffffff, then no such ICB is specified.

Table 1 - Virtual Allocation Table structure

Offset	Contents
0	LBA of virtual sector 0 [1/7.1.5]
4	LBA of virtual sector 1 [1/7.1.5]
8	LBA of virtual sector 2 [1/7.1.5]
...	...
2048	LBA of virtual sector 512 [1/7.1.5]
...	...
N * 4	Entity Identifier (regid) [1/7.4]
N * 4 + 32	Previous VAT ICB location [1/7.1.5]

The number of entries in the table can be determined from the VAT file size in the ICB:

Number of entries (N) = $\frac{FileSize - 36}{4}$

An entry of 0xffffffff indicates that the virtual sector is currently unused.

The LBA specified is located in the partition identified by the partition map.

5.3.2 Virtual Partition Map

The Logical Volume Descriptor contains a list of partitions that make up a given volume. As the virtual partition cannot be described in the same manner as a physical partition, a Type 2 partition map defined below shall be used.

The Logical Volume Descriptor shall contain at least two partition maps. The first, partition 0, shall be recorded as a Type 1 partition map [3/10.7.2]. The second, partition 1, shall be recorded as a Type 2 partition map [3/10.7.3]. The format of this Type 2 partition map shall be as specified in Table 2.

Table 2 - Layout of Type 2 partition map for virtual partition

RBP	Length	Name	Contents
0	1	Partition Map Type	Uint8 [1/7.1.1] = 2
1	1	Partition Map Length	Uint8 [1/7.1.1] = 64
2	2	Reserved	#00 bytes
4	32	Partition Type Identifier	regid [1/7.4]
36	2	Volume Sequence Number	Uint16 [1/7.1.3]
38	2	Partition Number	Uint16 [1/7.1.3]
40	24	Reserved	#00 bytes

The Domain ID in the Logical Volume Descriptor and the File Set Descriptor shall be recorded as follows:

- Flags = 0
- Identifier = *UDF Virtual Partition (byte sequence 0x2a 0x55 0x44 0x46 0x20 0x56 0x69 0x72 0x74 0x75 0x61 0x6c 0x20 0x50 0x61 0x72 0x74 0x69 0x74 0x69 0x6f 0x6e)
- IdentifierSuffix is recorded as in UDF 2.1.4.3
- Volume Sequence Number = volume upon which the VAT and Partition is recorded
- Partition Number = an identification of a partition within the volume identified by the volume sequence number

5.3.3 CD UDF end of session data

A session is closed to enable reading by CD-ROM drives. The last complete session on the disc shall conform completely to ISO 13346 and have two AVDPs recorded. This shall be accomplished by writing data according to Table 3. Although not shown in the following example, the data may be written in multiple packets.

Table 3 - CD UDF end of session data

Count	Description
1	Anchor Volume Descriptor Pointer
255	Implementation specific. May contain user data, file system structures, and/or link areas.
1	VAT ICB.

The implementation specific data may contain repeated copies of the VAT and VAT ICB. Compatibility with drives that do not accurately report the location of the last sector will be enhanced.

Implementations shall ensure that enough space is available to record the end of session data.

5.4 Random Access File System Specific

Writing which conforms to this section of the standard shall be performed using fixed length packets. The drive or device driver shall perform read/modify/write (see 5.4.4) to enable the apparent writing of single 2k sectors. The packet length shall be set when the disc is formatted. The packet length shall be 32 sectors (64 kB)..

CD physical format does not provide for sparing or bad block mapping. The host shall maintain a list of defects on the disc. The defect list shall be contained in a stream of the root directory (see section 5.4.6). Sparing shall be managed by the host

5.4.1 Formatting

Formatting shall consist of writing a lead-in, user data area, and lead-out. These areas may be written in any order. This physical format may be followed by a verification pass. Defects found during the verification pass shall be enumerated in the Defect Table Extended Attribute (see 5.5.4). Finally, file system root structures shall be recorded. These mandatory file system and root structures include the Volume Recognition Sequence[2/8.3.1], Anchor Volume Descriptor Pointers[3/10.2], a Volume Descriptor Sequence[3/8.4.2], a File Set Descriptor[4/14.1] and a Root Directory [4/14.1.15].

The Anchor Volume Descriptor Pointers shall be recorded at sectors 256 and N - 256, where N is the Physical Address of the last addressable sector.

Allocation for sparing shall occur during the format process. The sparing allocation may be zero in length.

The free space descriptors shall be recorded and shall reflect space allocated to defective areas and sector sparing areas.

The format may include all available space on the medium. However, if requested by the user, a subset may be formatted to save formatting time. That smaller format may be later “grown” to the full available space.

5.4.1.1 Growing the Format

If the medium is partially formatted, it may be later grown to a larger size. This operation consists of:

1. Optionally erase the lead-in of the last session.
2. Optionally erase the lead-out of the last session.
3. Write packets beginning immediately after the last previously recorded packet.
4. Update the sparing table to reflect any new spare areas
5. Adjust the partition map as appropriate
6. Update the free space map to show new available area
7. Move the last AVDP to the new N - 256
8. Write the lead-in (which reflects the new track size)
9. Write the lead-out

.

5.4.2 Sparable Partition Map

CD-E systems do not perform defect management. To provide an apparent defect-free space, a partition of type 2 is used. The partition map defines the partition number, packet size, and size and locations of the sparing tables. A partition descriptor shall be recorded as it is for a type 1 partition map.

Table 4 - Layout of Type 2 partition map for sparable partition

RBP	Length	Name	Contents
0	1	Partition Map Type	Uint8 [1/7.1.1] = 2
1	1	Partition Map Length	Uint8 [1/7.1.1] = 64
2	2	Reserved	#00 bytes
4	32	Partition Type Identifier	regid [1/7.4]
36	2	Volume Sequence Number	Uint16 [1/7.1.3]
38	2	Partition Number	Uint16 [1/7.1.3]
40	2	Packet Length	Uint16 [1/7.1.3] = 32
42	1	Number of Sparing Tables (=N_ST)	Uint8 [1/7.1.1]
43	1	Reserved	#00 byte
44	4	Size of each sparing table	Uint32 [1/7.1.5]
48	4 * N_ST	Locations of sparing tables	Uint32 [1/7.1.5]
48 + 4 * N_ST	16 - 4 * N_ST	Pad	#00 bytes

- Partition Type Identifier = *UDF Sparable Partition (byte sequence 0x2a 0x55 0x44 0x46 0x20 0x53 0x70 0x61 0x72 0x61 0x62 0x6c 0x65 0x20 0x50 0x61 0x72 0x74 0x69 0x74 0x69 0x6f 0x6e)
- IdentifierSuffix is recorded as in UDF 2.1.4.3
- Partition Number = the number of this partition. Shall identify a Partition Descriptor associated with this partition.
- Packet Length = the number of user data blocks per fixed packet. Shall be set to 32.
- Number of Sparing Tables = the number of redundant tables recorded. This shall be a value in the range of 1 to 4.

- Size of each sparing table = Length, in bytes, of each sparing table.
- Locations of sparing tables = the start locations of each sparing table specified as a media block address. Implementations should align the start of each sparing table with the beginning of a packet. Implementations should record at least two sparing tables in physically distant locations.

5.4.3 Host Based Defect Management

The host shall perform defect management operations. The CD format was defined without any defect management; to be compatible with existing technology and components, the host must manage defects. There are two levels of defect management: Marking bad sectors at format time (see 5.4.6) and on-line sparing (see 5.4.2 and 5.4.5). The host shall keep the tables on the media current.

5.4.4 Read Modify Write Operation

CD-E media requires large writable units as each unit incurs a 14KB overhead. The file system requires a 2KB writable unit. The difference in write sizes is handled by a read-modify-write operation by the host. An entire packet is read, the appropriate portions are modified, and the entire packet written to the CD.

Packets may not be aligned to 32 sector boundaries. Please see section 5.4.6.

5.4.5 Sparing Table

Sparing Tables point to space allocated for sparing and contains a list of mappings of defective sectors to their replacements. Separate copies of the sparing tables shall be recorded in separate packets. All sparing tables shall be kept up to date.

Each sparing table shall be structured as in Table 5.

Table 5 - Sparing Table layout

BP	Length	Name	Contents
0	16	Descriptor Tag	tag [3/7.2] = 0
16	32	CD-E Sparing Identifier	regid [1/7.4]
48	2	Reallocation Table Length (=RT_L)	Uint16 [1/7.1.3]
50	2	Reserved	#00 bytes
52	4	Sequence Number	Uint32 [1/7.1.5]
56	8*RT_L	Map Entry	Map Entries

This structure may be larger than a single sector if necessary.

- Descriptor Tag
  Contains 0, indicating that the contents are not specified by ISO 13346.

- CD-E Sparing Identifier
  Flags = 0
  Identifier = * UDF Sparing Table (byte sequence 0x2a 0x55 0x44 0x46 0x20 0x53 0x70 0x61 0x72 0x69 0x6e 0x67 0x20 0x54 0x61 0x62 0x6c 0x65)
  IdentifierSuffix is recorded as in UDF 2.1.4.3
- Reallocation Table Length
  Indicates the number of entries in the Map Entry table.
- Sequence Number
  Contains a number that shall be incremented each time the sparing table is updated.
- Map Entry
  A map entry is described in Table 6. Maps shall be sorted in ascending order by the Original Location field.

Table 6 - Map Entry description

RBP	Length	Name	Contents
0	4	Original Location	Uint32 [1/7.1.5]
4	4	Mapped Location	Uint32 [1/7.1.5]

- Original Location
  Logical Address of the packet to be spared. If this field is 0xffffffff, then this entry is available for sparing. The address of a packet is the address of the first user data block of a packet.
- Mapped Location
  Physical Address of active data. Requests to the original packet location are redirected to the packet location identified here. All Mapped Location entries shall be valid, including those entries for which the Original Location is 0xffffffff

5.4.6 Defect List

The defect list shall be generated at format time. All space indicated by the defect list shall also be marked as allocated in the free space map. The defect list shall be recorded as a file stream of the root directory. The stream name “Bad Sectors” shall be used. The name may be recorded in any legal word size. The information length of this stream shall be zero. This stream shall have all bad sectors identified by its allocation extents. The allocation extents shall indicate that each extent is allocated but not recorded.

5.4.7 Levels of Compliance

5.4.7.1 Level 1

The disc shall be formatted with exactly one lead-in, program area, and lead-out. The program area shall contain exactly one track. The start of the partition shall be on a packet boundary. The partition length shall be an integral multiple of the packet size.

5.4.7.2 Level 2

The last session shall contain the CD UDF file system. All prior sessions shall be contained in a single read-only partition.

5.4.7.3 Level 3

No restrictions shall apply.

5.5 Extended Attributes (Sequential and Random Access file systems)

The extended attributes described below are all structured to use the ISO 13346 Implementation Use Extended Attribute Figure [4/35]. Extended attributes may be embedded in the corresponding ICB, be described by an ICB identified by the parent file or directory ICB, or both. UDF already defines several implementation use identifiers for special uses, OS/2, and Macintosh.

Table 7 defines several extended attributes as an extension to the table in section 6.1 of UDF:

Table 7 - CD UDF extended attributes

Entity Identifier	Identifier Sequence (hex)	Type	Description
*UDF Path Table	2a 55 44 46 20 50 61 74 68 20 54 61 62 6c 65	Optional	Contains a path table describing the directory hierarchy on the disc.
*UDF Hash Table	2a 55 44 46 20 48 61 73 68 20 54 61 62 6c 65	Optional	Contains a hash table to improve performance of searches in large directories
*UDF Packet Table	2a 55 44 46 20 50 61 63 6b 65 74 20 54 61 62 6c 65	Optional	Contains a list of all packets recorded on the CD. Used for Sequential Access.
*UDF Defect Table	2a 55 44 46 20 44 65 66 65 63 74 20 54 61 62 6c 65	Optional	Contains a table of extents that identify defects on the media. Used for Random Access.

5.5.1 Path Table

The path table is a method for quickly finding a directory in the hierarchy. It contains a list of all directories and the location of their ICBs. The path table shall be recorded as an extended attribute of the root directory. Implementations that do not support a path table shall delete this extended attribute if the directory structure is modified.

The extended attribute shall be formatted as in Table 8.

Table 8 - Path Table Extended Attribute

RBP	Length	Name	Contents
0	48	Implementation Use Extended Attribute	Figure [4/35]
48	2	Header Checksum	Uint16 [1/7.1.3]
50	1	Flags	Uint8 [1/7.1.1]
51	1	Reserved	#00 byte
52	4	Number of Records	Uint32 [1/7.1.5]
56	*	Path Table Records	See Table 9

- Implementation Use Extended Attribute
  See section [4/14.10.8]. The attribute length shall be a multiple of 4.
  The identifier shall be *OSTA UDF Path Table.
- Header Checksum
  See UDF 3.3.4.5
- Flags
  Bit 0 is the “sorted” flag. If clear, the records in the table are not sorted. If set, the records shall be sorted first by level in the hierarchy and second by their file identifiers (using a bytewise comparison). The first entry shall be for the root directory and shall indicate the root directory as its parent.
  Bits 1-7 are reserved for future use.
- Number of Records
  Shall specify the number of records contained in the path table.
- Path Table Records
  A series of path table records. The first record shall be for the root directory.

Table 9 - Path Table Record layout

RBP	Length	Name	Contents
0	4	Directory Number	Uint32 [1/7.1.5]
4	4	Parent Directory Number	Uint32 [1/7.1.5
8	6	Location of Directory lCB	lb_addr [4/7.1]
14	1	Length of File Identifier (=L_FI)	Uint8 [1/7.1.1]
15	1	Flags	Uint8 [1/7.1.1]
16	L_FI	File Identifier	OSTA Compressed Unicode
16 + L_FI	*	Padding	#00 bytes

- Directory Number
  The record number of this record in the set of path table records. The first entry shall be for the root directory and shall be Directory Number 1.
- Parent Directory Number
  The record number of the record that describes the parent of the current record.
- Location of Directory lCB
  A pointer to the lCB that describes the directory that the current record describes.
- Length of File Identifier
  Length, in bytes, of the File Identifier field
- Flags
  Bit 0 is the "Hash Table Present" flag. If clear, the directory does not have an associated hash table. If set, a hash table is recorded as an extended attribute of the directory.
  Bits 1-7 are reserved for future use.
- File Identifier
  Shall be written in OSTA compressed Unicode format. It shall have the same byte sequence as the name recorded in the parent directory's file identifier.
- Padding
  This field shall be $4 \times ip((16 + L\_FI + 3) / 4) - (16 + L\_FI)$ bytes long and shall contain all #00 bytes.

5.5.2 Directory Hash Table

A Directory hash table may be recorded as an extended attribute of a directory. The hash table contains entries for all File Identifiers within that directory. The hash table shall be a fixed length per directory. The hash table may be recorded for only some of the directories present. The hash table becomes effective for directories with greater than 160 directory entries. Implementations that do not support a hash table shall delete this extended attribute if its associated directory is modified.

Implementations that use a hash table may use the deleted bit of file identifier entries to indicate deleted files. Implementations that remove file identifier entries to delete a file shall adjust all hash table entries appropriately.

The hash table shall be initialized to contain all 0.

The hash table uses the open addressing method with double hashing method (also called rehashing or linear quotient hashing). This method utilizes two hash functions, one for accessing the primary position of the key, *h*, and a second function, *p* for resolving conflicts.

The table size, *TSize*, should be a prime number so that each position in the table can be included in the sequence. The probing sequence becomes:
[h(k) - i*p(k)] % TSize for i in the range 0 <= i <= (TSize - 1)
This is equivalent to:
h(k), h(k) - p(k), h(k) - 2p(k), ..., h(k) - (TSize - 1)*p(k)
where:
TSize = Size of the hash table
k = Key(FileIdentifier)
h(k) = k % TSize
p(K) = max (1, k/TSize)

Tsize should not be a prime of the form:
$Tsize = r^k \pm a$
where:
r is the radix of the character set (256)
K the key
k and a small integers, like the number called a Fermat prime ($65537 = 2^{16} + 1$)

The resulting hash key shall be an index into a table. The table shall have TSize entries and shall be TSize * 4 bytes long.

The extended attribute shall be formatted as in Table 10.

Table 10 - Hash Table Extended Attribute layout

RBP	Length	Name	Contents
0	48	Implementation Use Extended Attribute	Figure [4/35]
48	2	Header Checksum	Uint16
50	2	Number of hash entries (=TSize)	Uint16
52	4	Offset for index 0	Uint32
56	4	Offset for index 1	Uint32
...	...	...	...
TSize * 4 + 48	4	Offset for index TSize - 1	Uint32

- Implementation Use Extended Attribute
  See section [4/14.10.8]. The attribute length shall be 52 + TSize * 4.
  The Implementation Use Length (=IU_L) shall be TSize * 4 + 4.
- Header Checksum
  See UDF 3.3.4.5
- Number of hash entries
  The number of entries in the hash table (TSize)
- Offset for index 0
  The offset, in bytes, from the beginning of the directory file to the FID that corresponds to hash index 0.
- Offset for index 1
  The offset, in bytes, from the beginning of the directory file to the FID that corresponds to hash index 1.
- Offset for index TSize - 1
  The offset, in bytes, from the beginning of the directory file to the FID that corresponds to hash index TSize - 1.

The algorithm in Figure 1 is used to generate a key from a file identifier if 8 bit identifiers are used:

Figure 1 - Key generation algorithm

```
INT32 Key ( char * FileName )
{
   INT32 ky = 0, g;

   while (*FileName) {
      ky = (ky << 4) + toupper(*FileName)
      FileName++
      g = ky & 0xf0000000;
      ky ^= g >> 24;
      ky &= ~g;
   }
   return ky;
}
```

Figure 2 is example code to add a File Identifier address into the hash table:

Figure 2 - Sample code for insertion of a new entry into the Hash Table

```
BOOLEAN InsertIntoHashTable (char *FileName, UINT32 Offset )
// Add The FID address into the Table
// T is the hash table, represented as an array of UINT32
{
   INT32 h, p, ht;  // the two hash functions
   int i = 0;

   h = ht = Key(FileName) % Tsize;
   p = max ( 1, Key(FileName) / Tsize);

   while ( T[ht].Key ) {
      i++;
      if (TSize == i )  {      // Table is Full
         return (FALSE);
      }
      ht = (h - i*p) % TSize;
      if ( ht < 0 ) {
         ht += TSize;
      }
   }
   T[ht].Key = Offset;
   return TRUE;
}
```

The function in Figure 3 is an example of how to use the hash table to find a file identifier.

Figure 3 - Hash Table search algorithm

```
LONG  SearchIntoHashTable (char * SearchFileName )
// Search if a file exists and return the FID address, otherwise NULL
{
   INT32 h, p, ht;        // the two hash functions
   int i;

   h = ht = Key(FileName) % TSize;
   p = max ( 1, Key(FileName) / Tsize);

   while ( T[ht].Key ) {
      if ( !strcmp (SearchFileName, (char *) UDFFileName(T[ht].Key) ) {
         return  T[ht].Key;
       }
       i++;
      if (TSize == i )  {
         TableFull = TRUE;
         return (NULL);
       }
      ht = (h - i*p) % TSize;
      if ( ht < 0 ) {
         ht += TSize;
       }
   }
   return NULL;
}
```

The UDFFileName(T[ht].key) is a function that when given the FID address returns the File ID (file name);

5.5.3 Packet Table Extended Attribute

A packet table may be recorded as an extended attribute of the VAT. The packet table lists the location and the size of every packet on the disc. This information may be necessary to bypass bugs in certain file system drivers with a read-ahead cache. Certain read-ahead implementations may require the use of a packet table to enhance performance. Implementations that do not support a packet table shall clear the “current” bit if the disc is updated.

The Packet Table only applies to variable packet written media.

The extended attribute shall be formatted as in Table 11.

Table 11 - Packet Table Extended Attribute layout

RBP	Length	Name	Contents
0	48	Implementation Use Extended Attribute	Figure [4/35]
48	2	Header Checksum	Uint16
50	1	Flags	Uint8
51	1	Padding	#00 byte
52	2	Number of packet entries (=N)	Uint32
56	4	Start address of packet 0	Uint32
60	4	Length of packet 0	Uint32
...	...	...	...
N * 8 + 48	4	Start address of packet N - 1	Uint32
N * 8 + 52	4	Length of packet N - 1	Uint32

- Implementation Use Extended Attribute
  See section [4/14.10.8]. The attribute length shall be 56 + N * 8.
- Header Checksum
  See UDF 3.3.4.5
- Flags
  Bit 0 (current):
  If clear, this bit shall indicate that the packet table may not be current.
  If set, the packet table is current.
  Bits 1-7 are reserved.
- Number of packet entries
  The number of entries in the packet table (N)
- Start address of packet 0
  The Physical Address of the first valid data sector of packet 0.
- Length of packet 0
  The number of valid data sectors in packet 0.

- Start address of packet N - 1
  The Physical Address of the first valid data sector of packet N - 1.
- Length of packet N - 1
  The number of valid data sectors in packet N - 1.

5.5.4 Defect Table Extended Attribute

The defect table is a method for identifying defective areas on a medium. Defective areas are marked as allocated but do not have any file system references to that area. The defect table is only needed during a file system check, as the check would normally free any space that is allocated but not referenced. The Defect Table Extended Attribute shall be recorded as an extended attribute of the root directory.

This extended attribute applies only to the Random Access method.

The extended attribute shall be formatted as in Table 12.

Table 12 - Defect Table Extended Attribute layout

RBP	Length	Name	Contents
0	48	Implementation Use Extended Attribute	Figure [4/35]
48	2	Header Checksum	Uint16 [1/7.1.3]
50	2	Partition Number	Uint16 [1/7.1.3]
52	2	Number of Extents (=N_E)	Uint16 [1/7.1.3]
54	2	Reserved	#00 bytes
56	8	Defective Extent 0	short_ad [4/14.14.1]
64	8	Defective Extent 1	short_ad [4/14.14.1]
72	...	...	...
48 + 8*N_E	8	Defective Extent N_E -1	short_ad [4/14.14.1]

- Implementation Use Extended Attribute
  See section [4/14.10.8]. The attribute length shall be a multiple of 4.
  The identifier shall be *CD UDF Defect Table.
- Header Checksum
  See UDF 3.3.4.5
- Partition Number
  The partition number to which this table applies.
- Number of Extents
  The number of extents described in this table
- Defective extent n
  Describes, in logical sectors (relative to the partition), locations of defective areas on the medium.

6. Multisession and Mixed Mode

The Volume Recognition Sequence [2/8.3.1] and Anchor Volume Descriptor Pointer locations are specified by ISO 13346 to be at a location relative to the beginning of the disc. The beginning of a disc shall be determined from a base address s for the purposes of finding the VRS and AVDP.

's' is the Physical Address of the first data sector in the first recorded data track in the last existent session of the volume. 's' is the same value currently used in multisession ISO 9660 recording. The first track in the session shall be a data track.

'n' is the physical sector number of the last recorded data sector on a disc.

If random write mode is used, the media may be formatted with zero or one audio sessions followed by exactly one writable data session containing one track. Other session configurations are possible but not described here. There shall be no more than one writable partition or session at one time, and this session shall be the last session on the disc.

6.1 Volume Recognition Sequence

The following descriptions are added to the CD UDF (see also ISO/IEC 13346 Part 2) in order to handle a multisession disc.

The volume recognition area [2/8.3] of the CD UDF Bridge format shall be the part of the volume space [2/8.2] starting at sector s + 16.

The volume recognition space shall end in the track in which it begins. As a result of this definition, the volume recognition area [2/8.3] always exists in the last session of a disc.

When recorded in Random Access mode, a duplicate Volume Recognition Sequence shall be recorded beginning at sector n - 256.

6.2 Anchor Volume Descriptor Pointer

Anchor Volume Descriptor Pointers shall be recorded at the following logical sector numbers: s + 256 and n - 256. The AVDP at sector n - 256 shall be recorded before closing a session; it may not be recorded while a session is open.

6.3 CD UDF Bridge format

The UDF Bridge format allows CD UDF to be added to a disc that may contain another file system. A CD UDF Bridge disc shall contain a CD UDF file system in its last session. The last session shall follow the rules described in "Multisession and Mixed Mode" section above.

The disc may contain sessions that are based on ISO 9660, audio, vendor unique, or a combination of file systems. The CD UDF Bridge format allows CD enhanced discs to be created.

The CD UDF session may contain pointers to data in other sessions, pointers to data only within the CD UDF session, or a combination of both.

Some examples of CD UDF bridge discs are shown in Figures 4 - 7.

Figure 4 - Multisession CD UDF disc

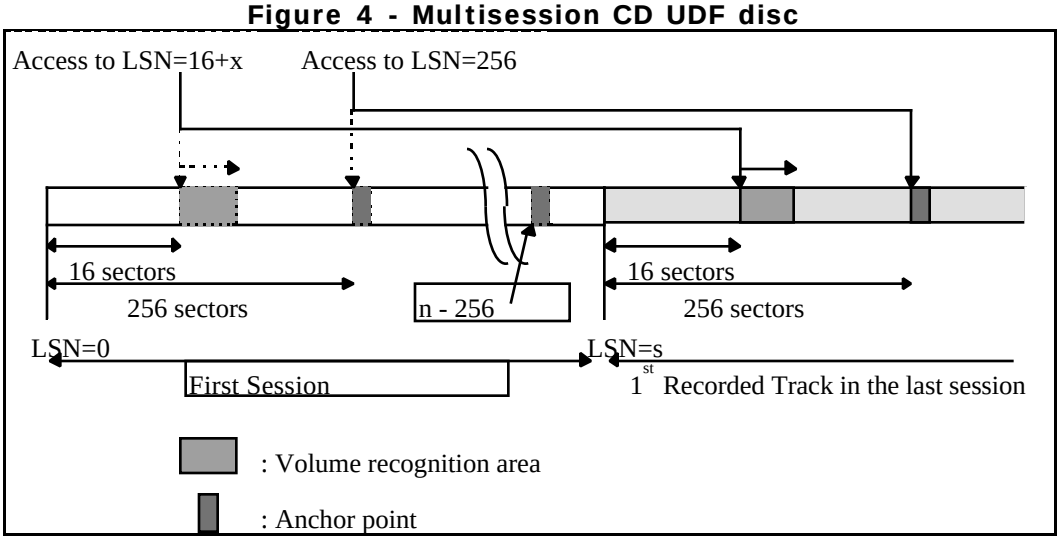


Figure 5 - CD enhanced disc

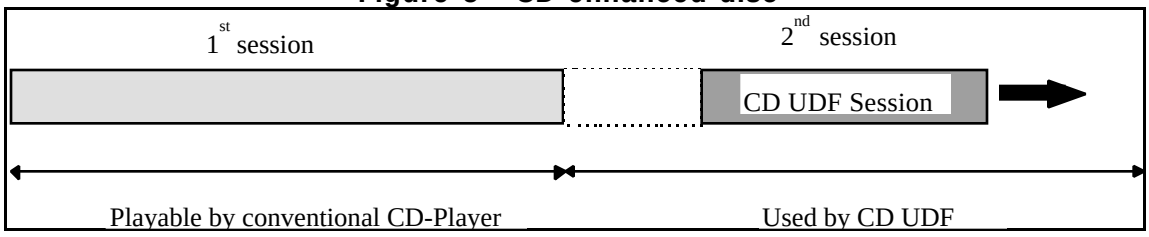


Figure 6 - ISO 9660 converted to CD UDF

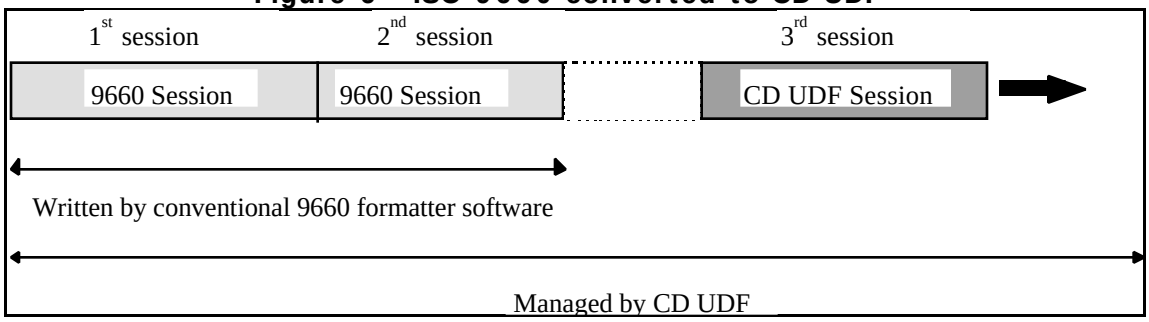
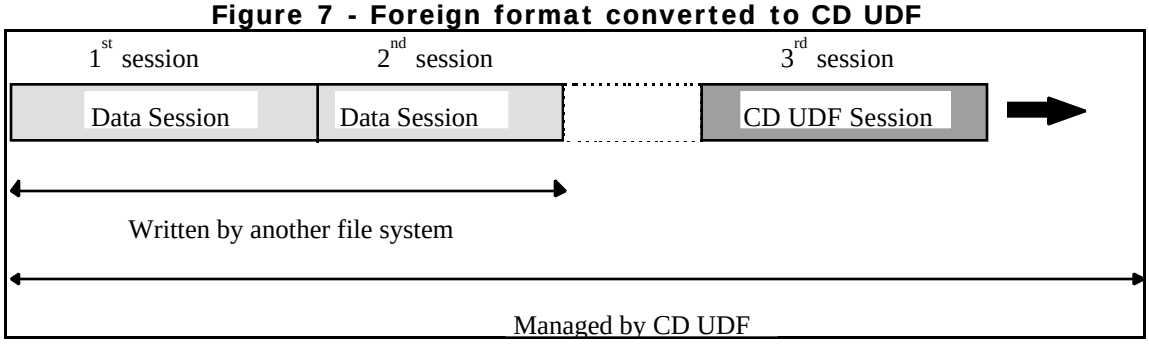


Figure 7 - Foreign format converted to CD UDF



7. Sequential Access Implementation Strategies

7.1 Allocation of space

The CD UDF file system allows at least three strategies for arranging information on the CD:

1. All information is written to a single track (the invisible track)
2. A track is reserved at the beginning of the free space. All file and directory ICBs are written in the reserved track and all file data is written to the data (invisible) track.
3. Two tracks are reserved. The first track will eventually hold the ISO 9660 file system, the second track is used for file and directory ICBs, and the invisible track holds the file data.

7.1.1 Strategy 1

Strategy 1 is the most space efficient when used as the only file system. It is also the simplest. However, closing to ISO 9660 requires closing the current session and writing an additional session with ISO 9660 in it.

The initialization of the disc shall consist of writing the ISO 9660 PVD, the ISO 13346 AVDP and the required supporting structures. A complete ISO 9660 structure shall be written. The file set shall contain a text file identifying the disc as a CD UDF disc. It may also contain drivers to help interchange with other systems. All writing occurs in the unreserved last track.

7.1.2 Strategy 2

Strategy 2 is similar to strategy 1 except that performance gains will be seen due to the clustering of the file system information in a small physical area.

If the file system area becomes full, the file system track shall be closed. The invisible track shall be closed. A new file system track shall be reserved. Writing may then continue into the new reserved file system track and file data into the invisible track.

7.1.3 Strategy 3

Strategy 3 provides a higher degree of compatibility with ISO 9660. The first track is reserved for track at once recording, which is filled in when the session is closed. However, due to minimum track lengths, the ISO 13346 Anchor Volume Descriptor Pointer cannot be initially recorded at sector 256. It shall be recorded at sector 512. The AVDP shall also be recorded in sector 256 when the file system is closed.

7.2 Duplicate VAT

The VAT is always recorded at the end of the invisible track. However, a VAT ICB may also be recorded at the end of a separate file system track. This second VAT ICB shall identify the VAT ICB in the data area as its parent. VAT ICBs recorded in the data area shall identify the previous VAT ICB recorded in the data area as its parent. The duplicate VAT ICB may be used to check file system integrity or aid in error recovery.

A duplicate VAT shall be recorded if strategies one or two are implemented.

7.3 Duplicate ICB

Each set of file data shall have at least one ICB that identifies it. The logical and physical location of an ICB is not specified. The ICB may be recorded in the data area, the file system area, or both. If recorded in both areas, the ICB in the file system area shall be identified by the VAT and the ICB in the data area shall not appear in the file system hierarchy. If recorded in both areas, the data area ICB shall be used solely for error recovery procedures.

7.4 Closing a Session

Please see section 5.3.3.

8. Sequential File System Sample Sequence of Events

The following is an example of a normal sequence of events:

1. Format the disc:
   a) [strategies 2 and 3] Reserve track(s)
   a) [strategy 1] Write ISO 13346 volume recognition structures (which may include ISO 9660 structures)
   b) Record the AVDP [3/10.2], PVD [3/10.1], PD [3/10.5], LVD [3/10.6], etc.
   c) Record a root directory (which may be in the root directory ICB) and a root directory ICB.
   d) Record a VAT (which may be contained in the VAT ICB) and a VAT ICB.
2. Write file data. This data consists of any new information, either for a completely new file or new data within an existing file.
3. Write the file ICB [4/14.9]. This ICB describes the file, including a list of all extents of the file. If the extent list is sufficiently large to overflow the sector, the continued Allocation Extent Descriptor shall be recorded before this file ICB. This ICB shall be recorded in the file system track if strategies 2 or 3 are used.
4. Write the directory data (if not embedded in the directory ICB). The directory must be updated to point to the file's new ICB (and the file version number may also be incremented). Directory data shall be recorded in the file system track if strategies 2 or 3 are used.
5. Write the directory ICB [4/14.9]. The directory ICB shall be recorded in the file system track if strategies 2 or 3 are used.
6. Repeat from step 2 if large updates are desired.
7. Write the VAT data (if not embedded in the VAT ICB). The VAT needs to reflect the new Logical Block Addresses of the new or updated directories.
8. Write the VAT ICB to the data area. If strategies 2 or 3 are used, record a VAT ICB in the file system track. The disc is now in a clean state.
9. Repeat from step 2 as necessary.

To "close" the disc to ISO 9660 compatibility (Strategies 1 and 2)

1. Close the session.
2. Generate ISO 9660 image for existing directories.
3. Write a session containing the ISO 9660 image and CD UDF end of session data.
4. Close this new session.

To "close" the disc to ISO 9660 compatibility (Strategy 3)

1. Generate ISO 9660 image for existing directories.
2. Write the ISO 9660 image to the first reserved track.
3. Write CD UDF end of session data.
4. Close the session.

For disc insertion:

1. Read sector s + 256. This sector shall contain the Anchor Volume Descriptor Pointer, which identifies the Volume Descriptor Sequence. If sector s + 256 is not recorded, read sector 512. See “Multisession and Mixed Mode” section above.

2. Read the Volume Descriptor Sequence. It will contain at least the Logical Volume Descriptor and at least one Partition Descriptor.
   Cache the partition information and the File Set Descriptor location.
3. Obtain the "Next Writable Address" for the last (invisible) track.
   Subtract 8 from the NWA to get the Physical Address of the last written sector.
4. Read the last written sector.
   Verify that the sector contains an ICB for file type 0. Verify that the ICB describes a VAT by checking the identifier at the end of the VAT. If not a valid VAT, go to "Error Recovery."
5. Extract and cache the VAT. The VAT location will be identified by the VAT ICB.

Error Recovery:

1. Read sectors from (NWA - 8) to 257 until a VAT ICB (file type 0) is found. Verify that the ICB describes a file that has a VAT Entity Identifier at the end. This operation is more efficient if large blocks are read from the disc at one time.
2. Record a new VAT ICB at the end of the track. Advanced recovery techniques would examine all data following the found ICB for other types of ICB and recover those structures also.

9. Sequential File System Example Disc Images (strategy 1)

Table 13 - Sector Map for formatted "blank" disc

VSN	LSN	PSN	Description	General
-	-	0-15	Implementation specific, i.e. system boot area.	ISO 9660 Format Disc Init
-	-	16	ISO 9660 Primary Volume Descriptor	
-	-	17	ISO 9660 Volume Descriptor Set Terminator	
-	-	18	ISO 13346 Beginning Extended Area Descriptor	
-	-	19	ISO 13346 Volume Structure Descriptor "NSR02" [3/9.1]	
-	-	20	ISO 13346 Terminating Extended Area Descriptor	
-	-	21	Blank (may be data but must be non-descriptor)	
-	-	22-28	Link blocks (not needed if streaming can be guaranteed)	
-	-	29	ISO 9660 type L path table - describes root directory	
-	-	30	ISO 9660 type M path table - describes root directory	
-	-	31	ISO 9660 root directory - describes README.TXT and, optionally, other files, e.g. drivers	
-	-	32	Data for file "README.TXT"	
-	-	33-56	Data for file "CD_UDF.VXD"	
-	-	57-63	Link blocks (not needed if streaming can be guaranteed)	
-	-	64-95	Data for file "CD_UDF.IFS"	
-	-	96-102	Link blocks	
-	-	103	ISO 13346 Primary Volume Descriptor	ISO 13346 Format Disc Init
-	-	104	ISO 13346 Logical Volume Descriptor. This structure points to the FSD in partition 1 (the virtual partition), sector 0.	

-	-	105	ISO 13346 Partition Descriptor (partition 0). Partition spans physical sectors 257-333,000.
-	-	106	ISO 13346 Terminating Descriptor
-	-	107-113	Link blocks
-	-	114-248	Unused or more drivers
-	-	249-255	Link blocks
-	-	256	ISO 13346 Anchor Volume Descriptor Pointer (AVDP) Identifies the Volume Descriptor Sequence at sectors 103-106. This allows rewriting the VDS if an underrun occurs.
0	0	257	File Set Descriptor. Points to the root directory at partition 1 (the virtual partition), sector 1.
1	1	258	Root Directory ICB with embedded directory
-	2	259	VAT ICB with embedded map. Virtual Sector 0 is identified as logical sector 0 Virtual Sector 1 is identified as logical sector 1

1. VSN is the virtual Sector Number. This number is an index into the Virtual Allocation Table.
2. LSN is the logical Sector Number. This number is relative to the start of partition 0.
3. PSN is the physical Sector Number. This number is relative to the start of the disc (per Yellow/Orange Books). This is equivalent to the ISO 9660 Logical Block Number.

Links may occur at points other than indicated or may not appear at all. The volume identifier sequence and the data for each file shall be recorded contiguously.

Maximum compatibility is achieved if there are no links in sectors 0-256.

After the formatting, files can be added to the disc. To add the file FOO.TXT in the directory BAR, the disc image appears as follows. FOO.TXT is assumed to be 4 sectors long. The following example assumes no underruns occurred.

Table 14 - Sector Map for disc with added directory and file

VSN	LSN	PSN	Description	General

-	-	0-256	No change from previous table.	Old Stuff
0	0	257	File Set Descriptor. Points to the root directory at partition 1 (the virtual partition), sector 1. (no change)	
-	1	258	Root Directory ICB with embedded directory. *Note change: this is no longer virtual sector 1!* This ICB has been replaced by the one at sector 273.	
-	2	259	VAT ICB with embedded map. Virtual Sector 0 is identified as logical sector 0. However, as this is not the last VAT, it is ignored.	
-	3-9	260-266	Link Blocks	
-	10-13	267-270	Contents of file FOO.TXT	Add a file and directory
-	14	271	File ICB describing FOO.TXT in sectors 267-270	
2	15	272	Directory ICB for directory BAR which contains an entry for FOO.TXT. The directory information is embedded in the ICB.	
1	16	273	Directory ICB for the root directory. It contains an entry for the BAR subdirectory.	
-	17	274	VAT ICB with embedded map. Virtual Sector 0 is identified as logical sector 0 Virtual Sector 1 is identified as logical sector 16 Virtual Sector 2 is identified as logical sector 15	

Table 15 - Underrun while recording a file

Next, here's an example of a write in which an underrun occurs. File HP.DAT is being written, and it is 20M in size. The file is being written into the root directory.

VSN	LSN	PSN	Description	General
-	-	0-256	No change from previous table.	Old Stuff
0	0	257	File Set Descriptor. Points to the root	

			directory at partition 1 (the virtual partition) sector 1. (no change)	
-	1	258	Root Directory ICB with embedded directory.	
-	2	259	VAT ICB with embedded map. Virtual Sector 0 is identified as logical sector 0. However as this is not the last VAT it is ignored.	
-	3-9	260-266	Link Blocks	
-	10-13	267-270	Contents of file FOO.TXT	File and dir added last time
-	14	271	File ICB describing FOO.TXT in sectors 267-270	
2	15	272	Directory ICB for directory BAR which contains an entry for FOO.TXT. The directory information is embedded in the ICB.	
-	16	273	Directory ICB for the root directory. It contains an entry for the BAR subdirectory.	
-	17	274	VAT ICB with embedded map. Virtual Sector 0 is identified as logical sector 0 Virtual Sector 1 is identified as logical sector 16 Virtual Sector 2 is identified as logical sector 15	
-	18-24	275-281	Link Blocks	
-	25-8216	282-8473	Data for HP.DAT. An underrun occurs after writing the first 16M, so this is only the first part.	New file Added
-	8217-8223	8474-8480	Link Blocks	
-	8224-10271	8481-10528	Remaining 4M of HP.DAT	
-	10272	10529	The root directory. It contains an entry for the BAR subdirectory and an entry for HP.DAT. The entry for HP.DAT has two extents: 8192 sectors at logical sector 25 and 2048 sectors at logical sector 8224.	
1	10273	10530	Directory ICB for the root directory. It points to the root directory at sector	

			10272. The directory is not embedded in the ICB, which will happen when the directory gets too large to fit.	
-	10274	10531	VAT ICB with embedded map. Virtual Sector 0 is identified as logical sector 0 Virtual Sector 1 is identified as logical sector 10273 Virtual Sector 2 is identified as logical sector 15	

10. Known incompatibilities between CD UDF and UDF

Table 16 - UDF and CD UDF known differences

Change from UDF	Type	Consequences
Only 1 Anchor Volume Descriptor Pointer is recorded instead of the required 2.	Sequential	The second AVDP is only for redundancy. Only conformance verification tools will find this difference. The final AVDP shall be recorded before closing the session, providing full compliance.
There will be references to a partition that does not exist.	Sequential	UDF implementations unaware of the VAT will not find the File Set Descriptor and will thus not find the file system.
There will be no allocation bitmap or list.	Sequential	UDF implementations unaware of CD UDF will not be able to write to CD-R media.
If space allocation strategy 3 is used the AVDP will be at sector 512.	Sequential	UDF implementations unaware of strategy 3 will not be able to mount the disc until it is "finalized" by writing the AVDP at sector 256.
Sparing tables added	Random Access	UDF implementations must be aware of the sparing table to remap sectors as necessary.
Parent ICB Location use	Both	The Parent ICB location points to another direct entry rather than an indirect entry. No known consequences.

Appendix A - Directory Hash Table Analysis

The number of accesses to find a directory entry depends on the size of the directory. This example assumes N entries in a directory and that the identifiers average 64 bytes in length (32 Unicode characters). This makes each file identifier descriptor 102 bytes long.

Number of directory sectors: $\frac{N \cdot 102}{2048}$ (Identifier average assumed to be 32 Unicode characters long)

Assume that a hash table needs < 10% occupied (sparse table); $10 \cdot N$ entries are needed; the table will be $\frac{40 \cdot N}{2048}$ sectors long.

Table 17 - Read analysis - hash table vs. no hash table

Item	No Hash Table	With Hash Table
Access to directory ICB	1	1
Access to directory sectors	1 - N * 102 / 2048 (20 identifiers/sector. Average seek is N * 51 / 2048 sectors)	N/A
Access to extended attribute ICB	N/A	1
Access to hash table (assume no or small other extended attributes)	N/A	1
Access to hash table entry	N/A	1
Access to directory to verify entry	N/A	1
If “collision” occurs	N/A	2

Assuming that there is no collision, the break even point is a five sector read. Given the 1 sector overhead without a hash table, if the average seek is 4 sectors long, the seek times are the same. 8 sectors hold 160 directory entries. The hash table at the break even point would be 5 sectors long.

Writing would incur an overhead in that every time a directory entry is updated, the hash table must also be updated. Updating the hash table involves rewriting the changed sector and rewriting the EA ICB. If the hash table is outgrown, the host must read the entire directory and rehash it within a new, larger table.

The average number of collisions encountered by 99% of all lookups can be determined from the following equation:

$$Collisions \le \frac{\log_{10}(0.01)}{\log_{10}\left(\frac{N}{Tsize}\right)}$$