-OSTA- ®

**Optical Storage
Technology Association**

UDF Compliance Test Disc Specification

Revision 1.0

July 8, 1999

REVISION HISTORY

0.00	July 23, 1998	Initial release, UDF 2.0 test cases only
0.01	August 20, 1998	Initial test case proposals added
0.02	September 23, 1998	ECMA 167 test cases added
0.03	October 23, 1998	ECMA 167 test cases added
		File Content Testing added
0.04	December 4, 1998	Test cases updated.
		Ready for final review
0.05	January 18, 1999	Adaptations made after final review
0.06	February 2, 1999	Small adaptations made before release
0.07	April 22, 1999	Final editorial review
0.99	May 12, 1999	Release candidate
0.995	June 7, 1999	Final draft for 1.0
1.0	July 8, 1999	Version 1.0

POINTS OF CONTACT

Optical Storage Technology Association

Ray Freeman
311 East Carrillo Street
Santa Barbara, CA 93101
Tel: +1 805 963 3853
Fax: +1 805 962 1541
Email: ray@osta.org
http://www.osta.org

OSTA Technical Reflector

Internet address for subscription: majordomo@hootie.lvld.hp.com
Internet address for distribution: osta_tc@hootie.lvld.hp.com

Technical Editor

Paul Deckers
Royal Philips Electronics
Email: time@lords.com

ABSTRACT

This specification defines the UDF Compliance Test Disc Specification for UDF. The applicable clauses of this specification containing the word "**shall**" are the requirements to be UDF Compliance Test Disc compliant. The Annexes are part of this document but are not required for compliance.

LICENSING

DISCLAIMER

IMPORTANT NOTICES

This document is a specification adopted by Optical Storage Technology Association (OSTA). This document may be revised by OSTA. It is intended solely as a guide for companies interested in developing products which can be compatible with other products developed using this document. OSTA makes no representation or warranty regarding this document, and any company using this document shall do so at its sole risk, including specifically the risks that a product developed will not be compatible with any other product or that any particular performance will not be achieved. OSTA shall not be liable for any exemplary, incidental, proximate or consequential damages or expenses arising from the use of this document.

This document is an authorized and approved publication of OSTA. The underlying information and materials contained herein are the exclusive property of OSTA but may be referred to and utilized by the general public for any legitimate purpose, particularly in the design and development of CD and DVD readable drives, systems and subsystems. This document may be copied in whole or in part provided that no revisions, alterations, or changes of any kind are made to the materials contained herein. Only OSTA has the right and authority to revise or change the materials contained in this document, and any revisions by any party other than OSTA are totally unauthorized and specifically prohibited.

Compliance with this document may require use of one or more features covered by proprietary rights (such as features which are the subject of a patent, patent application, copyright, mask work right or trade secret right). By publication of this document, no position is taken by OSTA with respect to the validity or infringement of any patent or other proprietary right, whether owned by a Member or Associate of OSTA or otherwise. OSTA hereby expressly disclaims any liability for infringement of intellectual property rights of others by virtue of the use of this document. OSTA has not and does not investigate any notices or allegations of infringement prompted by publication of any OSTA document, nor does OSTA undertake a duty to advise users or potential users of OSTA documents of such notices or allegations. OSTA hereby expressly advises all users or potential users of this document to investigate and analyze any potential infringement situation, seek the advice of intellectual property counsel, and, if indicated, obtain a license under any applicable intellectual property right or take the necessary steps to avoid infringement of any intellectual property right. OSTA expressly disclaims any intent to promote infringement of any intellectual property right by virtue of the evolution, adoption, or publication of this OSTA document.

INDEX

1 Introduction

This document defines a set of test media that will allow implementers of UDF file-system readers to test their applications for UDF compliance. The discs contain most if not all available data structures and options as described in the UDF specification. All data and structures on this disc are UDF compliant for it is beyond the scope of this specification to test how applications would cope with illegal structures.

This document does not describe a single test procedure. This is not feasible due to the many different types of implementations, each needing its own specific test suite. What this document does do is describe each specified structure that has been placed on the disc and how the reader should handle this particular item.

2 Disc set description

The disc set used for UDF compliance testing will consist of seven individual pieces of media containing a UDF file system and all files and directories needed to complete the tests as described in chapter 2. Chapter 3 describes how the files and directories are placed on the discs.

This chapter describes the general attributes of each disc in the set. These attributes can be defined as variable attributes and fixed attributes. The disc set was designed in such a way as to have the greatest number of permutations of the variable attributes as possible.

2.1   Variable attributes

The variable attributes have been applied to each disc as stated in the following table. Each of the specific attributes is described in detail in the next paragraphs.

	Disc 1	Disc 2	Disc 3	Disc 4
Sector Size	512 Bytes	2048 Bytes	2048 Bytes	2048 Bytes
ISO9660 bridge disc	NO	NO	NO	YES
Free space set	Table	None	Bitmap	None
UDF format	1.02	1.5 sequential	1.5 spared	2.0 sequential
Media type	WORM	CD-R (open)	CD-RW	CD-R (closed)
Multi Session	NO	YES	NO	NO
AVDP Location	256 & N	512	N-256 & N	256 & N-256
Strategy Type	4096	4	4	4
Streams	NO	NO	NO	YES

	Disc 5	Disc 6	Disc 7
Sector Size	2048 Bytes	2048 Bytes	2048 Bytes
ISO9660 bridge disc	NO	NO	YES
Free space set	Table	Bitmap	None
UDF format	2.0 random	1.02 random	1.5 sequential
Media type	CD-RW	CD-R (closed)	CD-R (closed)
Multi Session	NO	NO	YES
AVDP Location	256 & N	256 & N-256	256 & N-256
Strategy Type	4	4	4
Streams	YES	NO	NO

2.1.1   Sector sizes

The sector size is defined by the physical medium standard. For example, the sector size for mode 1 CD-ROM is always 2,048 bytes, even if the drive or its associated device driver reports a different sector size.

Notes:
A conforming implementation shall be capable of reading media having any of the allowed sector sizes for the medium types supported by the implementation, as follows:

Medium type	Supported Sector Sizes
CD-R	2048 bytes
CD-RW	2048 bytes
PD	512 bytes
5¼" WORM	512, 1024, 2048 bytes
5¼" rewritable	512, 1024, 2048 bytes
DVD-ROM	2048 bytes
DVD-R	2048 bytes
DVD-rewritable	2048 bytes
12"/14" WORM	1024, 2048, 8192 bytes
3½ " MO	512, 1024 bytes

The set of test discs described by this document includes discs with sector sizes of 512 bytes and 2048 bytes. The intention is to test the extremes of sector sizes on implementations that support 5¼" media, and not to imply that an implementation may choose not to support media with 1024 bytes per sector. CD format based implementations are not required to read test disc number 1.

2.1.2   ISO9660 Bridge

Discs 4 & 7 shall be ISO9660 bridge discs. The ISO9660 side of a bridge disc shall contain a single text file.

2.1.3   Free Space Set

Discs 1 & 5 shall use tables to define free space while discs 3 & 6 will use bitmaps.

2.1.4   UDF Format

The UDF formats used for the discs are: UDF 1.02, UDF 1.5 sequential, UDF 1.5 spared, UDF 2.0 sequential and UDF 2.0 random.

2.1.5   Media Type

The media types used for the discs are: WORM, CD-R (open), CD-R (closed) and CD-RW.

2.1.6   Multi Session

Discs 2 & 7 shall be recorded as multi-session disc. They shall both contain a closed first session and either an open (disc 2) or closed (disc 7) second session.

2.1.7   AVDP Locations

Each disc will have its AVDPs located according to the table in paragraph 2.1.

2.1.8   Strategy Type

Disc 1 shall use strategy 4096. Discs 2 to 7 shall use strategy 4.

2.1.9   Streams

Discs 4 and 5 shall contain streams.

2.2   Fixed attributes

2.2.1   Developer ID

The developer ID shall be for all discs:

*OITDA/OSTA UDF

The implementation Use Area of the Implementation Identifier Suffix shall be defined as:

Header	= #76#69 ("vi")
Major Revision Number	= 1 byte
Minor Revision Number	= 1 byte
Internal Build Number	= UINT16

The Major Revision Number, Minor Revision Number and Internal Build Number must contain version information of the tool used to create the disc content.

2.2.2   Copyright File

All discs shall include both a copyright file. The copyright file shall contain "UDF Specification Copyright Notice". The Copyright File Identifier shall be "copyright.txt". The Volume Copyright Notice extent (PVD) shall contain the same information as the copyright file.

2.2.3   Abstract File

All discs shall include both an abstract file. The abstract file shall contain "This is OSTA UDF test disc #", where # is the test disc sequence number as given in paragraph 2.1 . The Abstract File Identifier shall be "abstract.txt". The Volume Abstract extent (PVD) shall contain the same information as the abstract file.

2.2.4   Implementation Use Volume Descriptor

The Implementation Use Volume Descriptor shall contain the following information:

Owner:	UDF Compliance Subcommittee
Org:	OSTA
Contact:	http://www.osta.org

2.2.5   Entity ID

For the UDF identifier and implementation identifier the OS class version shall contain the appropriate value for the system creating the actual test discs.

2.3   Content Creator Information

Every disc shall contain in its root directory a file called "creatot.txt". The content of the file "creator.txt" is not specified in this document.

2.4   Multi Line Text files

All lines in a multi line text file shall be terminated by #0D, #0A.

3 Test Cases

This chapter describes each specific test case. Each case is defined using the following template:

Description	A short description of the case.
Discs	On which discs the case can be found.
Location on disc(s)	The full pathname where the specified test case can be found on the disc.
File names	The files used for the test
References	Reference to the ECMA and UDF standards
Rules	What should be expected from a compatible implementation

3.1   Time Zone

Description:

For each defined time zone, a file shall be recorded with a timestamp referring to that time zone. All files will be recorded with the same absolute time for all time stamps.

Discs:

All

Location on disc(s):

/Time/Time_Zones

File names:

GMT_-2400.TXT
GMT_-2330.TXT
GMT_-2300.TXT
GMT_-2200.TXT
GMT_-2100.TXT
GMT_-2000.TXT
GMT_-1900.TXT
GMT_-1800.TXT
GMT_-1700.TXT
GMT_-1600.TXT
GMT_-1500.TXT
GMT_-1400.TXT
GMT_-1300.TXT
GMT_-1230.TXT
GMT_-1200.TXT
GMT_-1130.TXT
GMT_-1100.TXT
GMT_-1000.TXT
GMT_-1000.TXT
GMT_-0900.TXT
GMT_-0800.TXT
GMT_-0700.TXT
GMT_-0600.TXT
GMT_-0500.TXT
GMT_-0400.TXT
GMT_-0300.TXT
GMT_-0200.TXT
GMT_-0100.TXT
GMT_-0030.TXT
GMT_-0015.TXT
GMT_-0005.TXT
GMT_-2047.TXT
GMT_0.TXT
GMT_+0005.TXT
GMT_+0015.TXT
GMT_+0030.TXT
GMT_+0100.TXT
GMT_+0200.TXT

```
GMT_+0300.TXT
GMT_+0400.TXT
GMT_+0500.TXT
GMT_+0600.TXT
GMT_+0700.TXT
GMT_+0800.TXT
GMT_+0900.TXT
GMT_+1000.TXT
GMT_+1100.TXT
GMT_+1130.TXT
GMT_+1200.TXT
GMT_+1230.TXT
GMT_+1300.TXT
GMT_+1400.TXT
GMT_+1500.TXT
GMT_+1600.TXT
GMT_+1700.TXT
GMT_+1800.TXT
GMT_+1900.TXT
GMT_+2000.TXT
GMT_+2100.TXT
GMT_+2200.TXT
GMT_+2300.TXT
GMT_+2330.TXT
GMT_+2400.TXT
```

References:

Applicable ECMA 167 references:	1/7.3
Applicable UDF 2.0 references:	2.1.4.1

Rules:

UDF requires that timestamps be recorded in local time. The time zone offset is specified in minutes from coordinated universal time and the value is either −2047, which means that no offset is specified, or in the range −1440 to +1440 minutes. This allows a range from −24 hours to +24 hours. Time zones West of Coordinated Universal Time have negative offsets.

When adjusted for time zone, the files in this directory all have the same date and time of creation. On systems that support time zones, all files should show the same creation and modification dates and times. On systems that do not support time zones, the times will be offset from the GMT_0.TXT time.

The creation and modification times for each file (after adjustment for time zone) is:

Year:	1998
Month:	8
Day:	7
Hour:	6
Minute:	5
Second:	4
Centisecond:	3
Hundreds of microseconds:	2
Microseconds:	1

3.2 Date Stress

Description:

21 files shall be recorded with timestamps in each decade beginning with decade starting in the year 1900 and ending with the decade starting in the year 2100.

Discs:

All

Location on disc(s):

/Time/Dates

File names:

"YEAR1900.TXT"
"YEAR1911.TXT"
"YEAR1922.TXT"
"YEAR1933.TXT"
"YEAR1944.TXT"
"YEAR1955.TXT"
"YEAR1966.TXT"
"YEAR1977.TXT"
"YEAR1988.TXT"
"YEAR1999.TXT"
"YEAR2000.TXT"
"YEAR2010.TXT"
"YEAR2021.TXT"
"YEAR2032.TXT"
"FEB6YEAR2040.TXT" => maximum time for HFS
"YEAR2054.TXT"
"YEAR2065.TXT"
"YEAR2076.TXT"
"YEAR2087.TXT"
"YEAR2098.TXT"
"YEAR2109.TXT"

References:

Applicable ECMA 167 references: 1/7.3
Applicable UDF 2.0 references: 2.1.4

Rules:

Implementations should display all timestamps to appropriate significance (as determined by OS requirements/limitations), or display an appropriate default or rounded timestamp as per the requirements and/or limitations of the OS. All files should be accessible.

3.3 Long File Names

Description:

Two files shall be recorded with file identifiers whose length shall be exactly 255 bytes in length with 8-bit Unicode compression (254 characters). The two filenames shall differ only by the last character of each filename.

Two files shall be recorded with file identifiers whose length shall be exactly 255 bytes in length with 16-bit Unicode compression (127 characters). The two filenames shall differ only by the last character of each filename.

Discs:

All

Location on disc(s):

/Filenames/Long_Filenames

File names:

"THIS_IS_A_LONG_8BIT_FILE_NAME _XXXXXX…….XXXXX_1.DAT"
"THIS_IS_A_LONG_8BIT _FILE_NAME_XXXXXX…….XXXXX_2.DAT"
"THIS_IS_A_LONG_16BIT_FILE_NAME _XXXXXX…….XXXXX_1.DAT"
"THIS_IS_A_LONG_16BIT _FILE_NAME_XXXXXX…….XXXXX_2.DAT"

References:

Applicable ECMA 167 references: 4/48.4.4
Applicable UDF 2.0 references: 2.1.1, 6.7

Rules:

The length of a file identifier is stored as a single byte. This gives a maximum length of 255 bytes for the representation of a file identifier. In UDF the identifier is stored as compressed Unicode and the first byte is reserved for a compression id. This leaves 254 bytes for the name itself. If the compression id is 8, each character is stored as a single byte and the name can be 254 bytes long. If the compression id is 16, each character is stored in two bytes and the name can be 127 characters long.

All four files should be accessible and be displayed with different names in accordance with the identifier translation algorithms for the operating system on which the disc is read.

3.4   Filename encoding

Description:

Six files shall be recorded using both 1-byte and 2-byte encoding of the filenames. Four files shall utilize non-roman character sets (e.g. Kanji).

One directory shall be recorded using 2-byte encoding using non-roman character sets (Kanji).

Discs:

All

Location on disc(s):

/Filenames/Filename_Encoding

File names:

File Name	Type	Encoding	File Name in hex
SINGLE_BYTE.TXT	File	1-Byte	"#53, #49, #4E, #47, #4C, #5F, #42, #59, #54, #45, #2E, #54, #58, #54"
DOUBLE_BYTE.TXT	File	2-Byte (Roman)	"#0044, #004F, #0055, #0042, #004C, #0045, #005F, #0042, #0059, #0054, #0045, #002E, #0054, #0058, #0054"
ユニバーサルディスク規定.TXT	File	2-Byte (Kanji)	"#30E6, #30CB, #30D0, #30FC, #30B5, #30EB, #30C7, #30A3, #30B9, #30AF, #898F, #5B9A, #002E, #0054, #0058, #0054"
1日本語.TXT	File	2-Byte (Roman/Kanji)	"#0031, #65E5, #672C, #8A9E, #002E, #0054, #0058, #0054"
え.TXT	File	2-Byte (Kanji)	"#3048, #002E, #0054, #0058, #0054"
日本語名.TXT	File	2-Byte (Kanji)	"#65E5, #672C, #8A9E, #540D, #002E, #0054, #0058, #0054"
日本語ディレクトリ	Directory	2-Byte (Kanji)	"#65E5, #672C, #8A9E, #30C7, #30A3, #30EC, #30EC, #30AF, #30AF"

References:

Applicable ECMA 167 references: NONE
Applicable UDF 2.0 references: 2.1.1, 4.2.2.1

Rules:

Implementation shall support Unicode character set standard.

If the characters within a file name are considered invalid or not displayable in the current environment, implementation shall translate them into "_" (#005F) character.

3.5   Stressed Filenames

Description:

A number of files and directories shall be recorded whose filenames should stress the receiving systems ability to handle unusual filenames.

Discs:

All

Location on disc(s):

/Filenames/Stressed

File names:

Test	Name	MS-DOS 8.3 Mangled name	Type	Comments
1	.	1.02: #C05 1.50: 4C05 2.00: C5AC 2.01: #U4A	Directory	
2	.	1.02: #C05 1.50: 4C05 2.00: C5AC 2.01: #U4A	File	To be placed in previous directory
3	..	1.02: #9E4 1.50: 49E4 2.00: 586B 2.01: #5VT	Directory	
4	..	1.02: #9E4 1.50: 49E4 2.00: 25E9 2.01: #5VT	File	To be placed in previous directory
5	.open me, please	1.02: #14A.OPE 1.50: 014A.OPE 2.00: 6A61.OPE 2.01: #0#6.OPE	Directory	
6	.open me, too, please	1.02: #260.OPE 1.50: D260.OPE 2.00: 9111.OPE 2.01: #TT0.OPE	File	To be placed in previous directory
7	Aa		File	
8	AA		File	
9	COMPID08		File	Encoded with compression id of 8
10	COMPID16		File	Encoded with compression id of 16
11	Open_me1.txt		File	
12	Open_me2.text	1.02: OPEN#D4D.TEX 1.50: OPEN6D4D.TEX 2.00: OPENF0DF.TEX 2.01: OPEN#M@J.TEX	File	
13	Open@me3		File	

Test	Name	MS-DOS 8.3 Mangled name	Type	Comments
14	Open:me4		File	
15	Open me5	1.02: OPEN#B9C 1.50: OPEN8B9C 2.00: OPEN841B 2.01: OPEN#JAT	File	First character is a blank
16	Here is a null char ><	1.02: HERE#C15 1.50: HERE5C15 2.00: HERE9E33 2.01: HERE#OQF	File	Between the >< is a (invisible) 0 character
17	\ /silly*Name?.	1.02: _SIL#9A9 1.50: _SIL329A9 2.00: _SIL1F38 2.01: _SIL#4U~	Directory	The first 2 characters are backslash and slash; not a capitol 'V'
18	*	1.02: _#C81 1.50: _0C81 2.00: _8528 2.01: _#KBH	File	
19	**	1.02: _#5A4 1.50: _C5A4 2.00: _C42F 2.01: _#ZK~	File	
20	??	1.02: _#BB6 1.50: _7BB6 2.00: _2E28 2.01: _#3CC	File	
21	_		File	Single underscore character
22	__		File	Double underscore character
23	Fold.t X	1.02: FOLD#EF1.TX 1.50: FOLDEEF1.TX 2.00: FOLDA6E1.TX 2.01: FOLD#9C9.TX	Directory	Character between t and X is a blank
24	File1.longextension	1.02: FILE#232.LON 1.50: FILE4232.LON 2.00: FILEB690.LON 2.01: FILE#DIT.LON	File	
25	Many.ext1.ext2.ext3	1.02: MANY#C65.EXT 1.50: MANYDC65.EXT 2.00: MANY664A.EXT 2.01: MANY#DJ~.EXT	File	
26	DOS< > : " / \ |.chars	1.02: DOS_#93B.CHA 1.50: DOS_D93B.CHA 2.00: DOS_1E68.CHA 2.01: DOS_#P59.CHA	File	Disallowed characters in DOS file names

References:

Applicable ECMA 167 references: NONE
Applicable UDF 2.0 references: 6.7

Rules:

Except for test 8 (deleted file) all files and directories should be accessible and should be named in accordance with the identifier translation algorithms for the operating system on which the disc is read.

3.6   Path Length

Description:

A number of directories shall be recorded in such a way that they shall create a directory path that reaches the limit of 1023 bytes.

Discs:

All

Location on disc(s):

/Deeptree

Path name:

/Deeptree/OSTA001/OSTA002/OSTA003/OSTA004/OSTA005/ ……
/OSTA125/OSTA126/EINDE

Note:

There are 126 embedded OSTA directories.

References:

Applicable ECMA 167 references: NONE
Applicable UDF 2.0 references: 2

Rules:

Implementations must be able to coherently handle directory path lengths that are longer than supported by the OS. By "coherently handle", it is meant that the implementation not crash, or take any destructive action.

It is not required that directory paths longer than maximum OS supported length be accessible.

3.7   Large Directory

Description:

A directory is recorded with 50000 files. The file names are the ASCII numbers from 1 through 50000. The content of each file is the name. For instance the file named "34" is two bytes in length and those two bytes are ASCII "3" and "4".

Discs:

All

Location on disc(s):

/Filenames/50K_Files

File names:

1
2
...
49999
50000

References:

Applicable ECMA 167 references: NONE
Applicable UDF 2.0 references: NONE

Rules:

All 50000 files should be accessible and correctly readable.

3.8   Large File

Description:

Two large files shall be recorded whose size 2GB-1 and 16GB in total using unallocated and unrecorded space extents.

Disc(s):

All

Location on disc(s):

/Large_File

Filenames:

/Large_File/2GB-1_FILE
/Large_File/16GB_FILE

Notes:

The first logical block of the file shall contain "START " followed by #00 bytes.
The last logical block of the file shall contain " END" preceded by #00 bytes.
All blocks in between are unrecorded and unallocated space.

Though the file size is 64-bit, other restrictions apply limiting the practical size of the file. So it was decided that we would test a file larger than the 32-bit limit (4GB).

References:

Applicable ECMA 167 references: 4/14.9.10
Applicable UDF 2.0 references: NONE

Rules:

Implementations should be able to read the 16GBFILE file if the operating system supports files larger than 4GB. On operating systems that support files larger than 4GB the file should be completely readable and the data should be verified as written in the notes section. If the operating system does not support files larger than 4GB they should either ignore the file or read it till the maximum size possible under the operating system.

2GB-1 file is created to test implementations that only support a 32-bit maximum file size. The behavior shall be similar to the 16GB file.

Note:

The 2GB-1 was created to support systems that are limited to 2GB file sizes.

3.9   Multiple Descriptors

Description:

A disc shall be recorded with multiple instances of a descriptor in the Volume Descriptor Sequence. The prevailing descriptor shall be placed before an earlier one.

Discs:

DISC	Multiple descriptor instances
D1	
D2	
D3	3 contiguous descriptors
D4	
D5	
D6	3 non contiguous descriptors
D7	

References:

Applicable ECMA 167 references: 3/8.4.3
Applicable UDF 2.0 references: NONE

Rules:

1. Disc 3: The non-prevailing Logical Volume Descriptors shall reference a non-existent partition descriptor of ID#4321.

2. Disc 6: 2 Partition Descriptors shall both reference the same partition: The non prevailing one has an incorrect starting block (starting block + 10) (partition #2)

3.10 Multiple LVIDs

Description:

A disc shall be recorded with multiple Logical Volume Integrity Descriptor instances (open and closed).

Discs:

DISC	Multiple LVID instances
D1	X
D2	
D3	
D4	
D5	
D6	X
D7	

References:

Applicable ECMA 167 references: 3/10.10
Applicable UDF 2.0 references: NONE

Rules:

LVD points to an extent of 1 block contains an open LVID that points to a second extent length 4 containing 3 LVID (closed, open, closed) followed by a terminating descriptor.

The number of files and folders in the first three both shall be 1. Only the last will contain the real number.

The free space table indicates that all blocks in the volume are free for the first three LVIDs. All other values should be compliant to spec.

3.11 Multiple FSDs

No test defined.

3.12 4096 Strategy

Description:

Read a subdirectory on the following discs of which a single file was recorded in strategy 4096.

Discs:

D2 – D7

Location on disc(s):

/Strategy_4096

File names:

4096_test1

References:

Applicable ISO reference(s):   14.6 - 14.8
Applicable ECMA 167 references: NONE
Applicable UDF 2.0 references: 2.3.5, 6.6

Note: See ISO specification for an explanation of terms.

Rules:

1: FE+IE -> FE+TE
2: FE+IE -> FE+IE->TE
3: FE+IE -> FE+BLANK
4: FE+IE -> FE+IE->BLANK

The instance of the file described by the 1st FE shall contain the string "incorrect file". The instance of the file described by the 2nd FE shall contain the string "correct file".

Implementation should show the correct file.

3.13 Partition Number

Description:

Partition Descriptor (and Partition Map Entry for type 1 partition): use a nonzero partition number for the type 1 partition, to see if an implementation correctly makes the distinction between partition number and partition reference number.

Discs:

All

Location on disc(s):

VDS: Partition Descriptor and VDS: Logical Volume Descriptor

Rules:

VDS:Partition Descriptor 1:Partition Number   = **X**
VDS:LVD:Partition Map N:Partition Number   = **X**

Values for **X** on specified discs are:

D1: X=101. D2: X=10007. D3: X=107. D4: X=10037. D5: X=113. D6: X=127. D7: X=131.

References:

Applicable ECMA 167 references: 3/10.5, 3/10.6
Applicable UDF 2.0 references: 2.2.4, 2.2.8, and 2.2.9

Notes:

- On discs D2, D4 and D7, Partition Descriptor 2 and Partition Map 2 describe the virtual partition.
- On disc D3, Partition Descriptor 2 and Partition Map 2 describe the sparable partition.
- Partition Descriptor 1 and Partition Map 1 describe the physical partition.
- VDS = Volume Descriptor Sequence.
- LVD = Logical Volume Descriptor.

3.14 ID Compression

Description:

Names in FIDs: include some deleted FIDs that use Compression IDs 8, 16, 254 and 255, and some non-deleted FIDs that use Compression IDs 8 and 16. The deleted files shall be before the non-deleted files in the directory.

Discs:

D4, D5

Location on disc(s):

/Filenames/Deleted_Files

File names:

CmpID_8
CmpID _16
CmpID _8_Deleted
CmpID _16_ Deleted
#00 #00 #00 #01
#00 #00 #00 #00 #00 #00 #00 #01

References:

Applicable ECMA 167 references: 4/8.6, 4/14.4.3 note 21, 4/14.4.5
Applicable UDF 2.0 references: 2.1.1, 2.3.4.2

Rules:

The application shall display all the non-deleted files' names correctly, and shall not display any of the deleted files. The contents of all non-deleted files shall be accessible.

3.15 Logical Volume Identifier

Description:

A Logical Volume Identifier consisting of unusual/illegal characters like null, '/', '\', ":", Back Space (in the first 11 characters) and whose length is more than 27 characters shall be recorded in the Logical Volume Descriptor.

Discs:

D3

Location on disc(s):

Logical Volume Descriptor

Name:

"OSTA\ /:<null><Back Space>&Logical Volume Identifier"

Please note that <null>, <Back Space> are single characters. Total length of the Identifier is 34 characters.

The 5th and 6th characters of this string are a '\' and a '/' character.

References:

Applicable ECMA 167 references: NONE
Applicable UDF 2.0 references: 4.1, 6.7

Rules:

Implementations should display the Identifier correctly, after using the Identifier Translation Algorithm (UDF 1.50, section 6.7) applicable to the relevant OS.

Notes:

The value of 27 is the maximum volume label length under MacOS.

3.16 Virtual & Non-virtual Space

Description:

Virtual Partition: have both ICBs and file data of both directories and files in both virtual and non-virtual space (using short allocation descriptors to verify that the right partition is referenced).

Discs:

All discs containing a VAT: D2, D4, and D7

Location on disc(s):

/ICBs_in_Space

File names:

Name	Type	Comments
F_vFE_rlAD	File	FE in VP, long AD, data in RP
F_vFE_sAD	File	FE in VP, embedded data in VP
F_rFE_rlAD	File	FE in RP, long AD, data in RP
F_rFE_sAD	File	FE in RP, short AD, data in RP
D_vFE_rlAD	Dir	FE in VP, long AD, data in RP
D_vFE_vlAD	Dir	FE in VP, embedded data in VP
D_rFE_rlAD	Dir	FE in RP, long AD, data in RP
D_rFE_sAD	Dir	FE in RP, short AD, data in RP

VP = Virtual Partition, RP = Real (Non-V) Partition

Notes:

Those files with data in the virtual partition must have their file data embedded in the file entry (UDF 2.3.10).

Rules:

Each directory contains a file called "In <X>", where X is the directory name, and the file content is always an ASCII text representing the files name, data is embedded in the FE, and the FE is always in the RP.

3.17 Sparing

Description:

Five files "Spared 1", "Spared 2", "Spared 3", "Spared 4" and "Spared 5" with sparing in various locations. The files "Not Spared 1" to "Not Spared 5", have content identical to that of "Spared 1" to "Spared 5".

Discs:

D3

Location:

/Spared_Files

Rules:

File "Spared 1" with 3 extents.

Extent1:
From packet boundary to packet boundary. Start at begin of Packet[D], continues in Packet[E], Packet[F] and Packet[G].
Packet[D] is spared inside the partition. Packet[F] is spared outside the partition between the VRS and sector 256. Packet[E] and Packet[G] are not spared.
Length: 262144 bytes
Extent2:
Start and end not at Packet boundary. Located over 2 Packets.
Extent starts at 32nd sector of packet[B], ends with the first sector of Packet[C]. Packet[B] is spared inside the partition.
Length: 4096 bytes
Extent3:
   The last sector of Packet [A] followed by sector 1 of Packet[B] (spared).
   Length 2084 bytes
Content:
   Each sector starts with a header that contains "/Spared files/Spared 1, Sector: n ". Where n=1 in the first sector of the file, and n increments by one throughout the file.
   In sectors 1..32, 65..96, 129 and 132 (the spared sectors), the bytes after the header contain "s". In sectors 33..64, 97..128, 130 and 131 (not spared sectors), the bytes after the header contain #20.

File "Spared 2" with one extent.

Extent:
   Sectors 2.31 of Packet [B] (spared).
   Length: 59428 bytes.
Content:
   Each sector starts with a header that contains "/Spared Files/Spared 2, Sector: n ". The bytes after the header contain "s". Where n=1 in the first sector of the file, and n increments by one throughout the file.

File "Spared 3" with one extent.
Extent:
    Start and End at a Packet boundary.
    Sector 1 in Packet[H] to sector 32 in Packet[J]. Packet[H] and Packet[J] are spared,
    Packet[I] is not spared.
    Length: 196608 bytes.
Content:
    Each sector starts with a header that contains "/Spared files/Spared 3, Sector: n ". Where
    n=1 in the first sector of the file, and n increments by one throughout the file. In sectors
    1..32 and 65..96 (the spared sectors), the bytes after the header contain "s". In sectors
    33..64 (not spared sectors), the bytes after the header contain #20.

File "Spared 4" with one extent.
Extent:
    Start and End spared, not at a Packet boundary.
    Sector 32 in Packet[K] to sector 1 in Packet[M]. Packet[K] and Packet[M] are spared,
    Packet[L] is not spared.
    Length: 69632 bytes.
Content:
    Each sector starts with a header that contains "/Spared files/Spared 4, Sector: n ". Where
    n=1 in the first sector of the file, and n increments by one throughout the file. In sectors 1
    and 34 (the spared sectors), the bytes after the header contain "s". In sectors 2..33 (not
    spared sectors), the bytes after the header contain #20.

File "Spared 5" with one extent.
Extent:
    Start and End not spared, not at a Packet boundary.
    Sector 32 in Packet[N] to sector 1 in Packet[P]. Packet[O] is spared, Packet[N] and
    Packet[P] are not spared.
    Length: 69632 bytes.
Content:
    Each sector starts with a header that contains "/Spared files/Spared 5, Sector: n ". Where
    n=1 in the first sector of the file, and n increments by one throughout the file. In sectors
    2..33 (the spared sectors), the bytes after the header contain "s". In sectors 1 and 34 (not
    spared sectors), the bytes after the header contain #20.

File "Dummy".
Extent1:
    Sectors 1..31 of Packet[A].
    Length: 63488 bytes.
Extent2:
    Sectors 2..32 of Packet[C].
    Length: 61440 bytes.
Content:
    Each starts with a header that contains "/Spared files/Dummy, Sector: n ". The bytes after
    the header contain "x". Where n=1 in the first sector of the file, and n increments by one
    throughout the file.

File "Not Spared 1".
Extent:
    No sparing.
    Length: 268324 bytes.
Content:
    A copy of the content of file "Spared 1".

File "Not Spared 2".
Extent:
    No sparing.
    Length: 59428 bytes
Content:
    A copy of the content of file "Spared 2".

File "Not Spared 3".
Extent:
    No sparing.
    Length: 196608 bytes
Content:
    A copy of the content of file "Spared 3".

File "Not Spared 4".
Extent:
    No sparing.
    Length: 69632 bytes
Content:
    A copy of the content of file "Spared 4".

File "Not Spared 5"
Extent:
    No sparing.
    Length: 69632 bytes
Content:
    A copy of the content of file "Spared 5".

Unallocated space
The following sectors are Unallocated:
- Sectors 1..31 in spared Packet[S]
- Sectors 2..32 in spared Packet[M]
- Sectors 1..31 in not spared Packet[N]
- Sectors 2..32 in not spared Packet[P]

Location of Packets

Used is a contiguous sequence of Packets, indexed from A to P. Packets B, D, H, J, K, M and O are spared inside the partition. Packet[F] is spared outside the partition between the VRS and sector 256. Packets A, C, E, G, I, L, N and P are not spared.
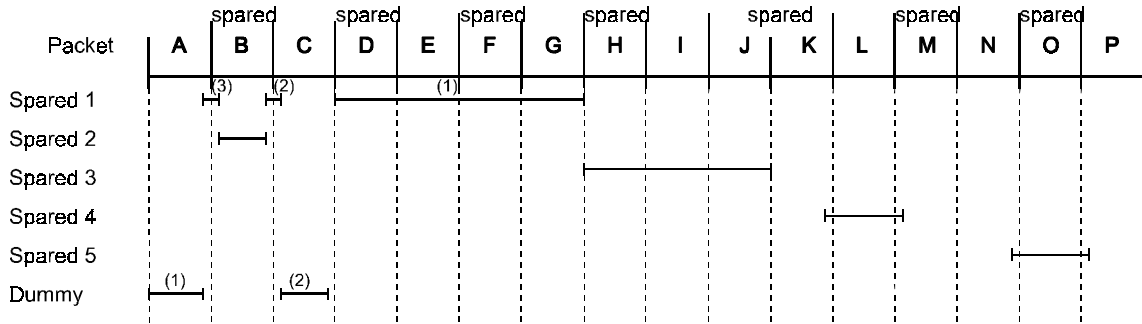
Content on original location

Each sector on the original location of the spared Packets starts with a header that contains "BAD sector P-N ", followed by bytes set to "b". Where P is the letter, that identifies the Packet (A..G) and n is the sequence number (starting at one) of the sector in the Packet.

References:

Applicable ECMA 167 references: NONE
Applicable UDF 2.0 references: UDF 2.2.9, UDF 2.2.11 and UDF 6.10.2.

3.18 Implementation Use field in FID

Description:

Check that a file containing an Implementation Use field can be read by the implementation. There may be an issue about very large Implementation Use fields pushing other fields over block boundaries.

Disks:

All

Location on disk(s):

/Impl_Use_Field_In_FID

Filename:

FID_BlockSpan
FID_FOLLOWS_BlockSpan

References:

Applicable ECMA 167 references: 4/14.4
Applicable UDF 2.0 references: 2.3.4,2.3.4.4-5

Rules:

FIDs total size is a single block size.
RegID is equal to that of the discs implementation ID
The rest is padded with #00
This FID will be followed directly by the FID for FID_FOLLOWS_BlockSpan

The conforming system shall read the FID_BlockSpan file, whose FID shall be spanning blocks, correctly.

3.19 File Types

Description:

The possible ICB tag file types are used. After the 14 predefined types the entry names are 14 through 255.

Discs:

All

Location on disc(s):

/Filetypes
/Filetypes/UDF2

File names:

unknown
directory
bytes
blockdevice
characterdevice
fifo
socket
symboliclink
/UDF2/TYPE248

Rules:

ICB Tag OSTA-2 Rev 2 section 4.2.1
File types ECMA 167 4/14.6.6
Unallocated ECMA 167 4/14.11
PIE ECMA 167 4/14.13
Indirect ECMA 167 4/14.7
Directory ECMA 167 4/8.6
Extended Attributes ECMA 167 4/9.1
Terminal Entry ECMA 167 4/14.8
Symbolic Link ECMA 167 4/8.7
Stream Directory ECMA 167 4/9.2

Minimal required operating system functionality is:

1) All names are visible in the directory.
2) Access to file types unsupported by the system shall be handled in a predictable manner *.

*Predictable manner to be defined by UDF

3.20 File Permissions

Description:

Permissions on directories and filenames for MS-DOS, OS/2, Windows 95, Windows NT, Macintosh, and UNIX: use OS neutral permissions to test a cross-platform compatibility and OS specific permissions to test an OS specific compatibility.

Discs:

All

Location on disc(s):

/PERM_OS   (OS specific files and directories)
/PERM_NON   (Non-OS specific files and directories)

File names:

PERM_NON Directory

FileCharacteristics bits in the File Identifier Descriptor (3.3.1.1) shall be set to the followings:

Bit	10	9	8	7	6	5	4	3	2	1	0
Value	0		0	0	0			0	0	0 or 1	0

Directories:

"re"   for read only - Read & Execute are set.
"rwe"   for read, write, and execute - Read, Write, Delete, & Execute are set.
"rwecd"   for read, write, execute, change attributes, and delete - All are set.

note: Permission bits are the same for Owner, Group, and Other.

Files:

All files listed bellow are in all three directories.

"r.txt"   for read only - Read is set.
"rwe.txt"   for read, write, and execute - Read, Write, Delete, & Execute are set.
"rwecd.txt"   for read, write, execute, change attributes, and delete - All are set.

note: Permission bits are the same for Owner, Group, and Other.

PERM_OS Directory

A directory will be created for each specific OS:

"UNIX"   for UNIX,
"DOS_WIN"   for MS-DOS, OS/2, Windows 95, and Windows NT
"MacOS"   for Macintosh.

The FileCharacteristics (3.3.2.1) bit in the File Identifier Descriptor (3.3.1.1) shall be set to ONE if it is supported by the specific OS. Uid and Gid should be set to "1000" if supported.

Directories:

Repeat the four sets of the following directories for the three possible settings - Owner only, Owner & Group, and Owner, Group, & Other.

"r"   for read only - Read is set.
"re"   for read and execute - Read & Execute are set.
"rwe"   for read, write, and execute - Read, Write, & Execute are set.
"rwecd"   for read, write, execute, change attributes, and delete - All are set.

Files:

Repeat the four sets of the following files for the three possible settings - Owner only, Owner & Group, and Owner, Group, & Other. All files listed bellow are in all three directories.

"r.txt"   for read only - Read is set.
"re.txt"   for read and execute - Read & Execute are set.
"rwe.txt"   for read, write, and execute - Read, Write, & Execute are set.
"rwecd.txt"   for read, write, execute, change attributes, and delete - All are set.

Rules:

PERM_NON

A receiving implementation should follow the permissions specified in each file and directory.

PERM_OS

A receiving implementation should support all the permissions and "FileCharacteristics" (3.3.1.1) for its OS specific directories and files and should follow the guidelines specified in the UDF specification section 3.3.3.3 for other OS specific directories.

3.21 Embedded Data

Description:

Embed file data and EAs into ICB (for both directories and data files).

Location on disc(s):

/Embedded_data

File names:

EMBED.TXT

Rules:

The Files FID is embedded in the directory.
The Files data is embedded in the ICB.

One must be able to correctly access the file.

3.22 Unrecorded Extents

Description:

Use unrecorded extents in the middle of a file, at the end of a file, and beyond the logical end of a file.

Location on disc(s):

/unrecorded/middle
/unrecorded/end
/unrecorded/beyond

File names:

/Unrecorded/Middle

This consists of 3 extents of 1 block each. The middle is marked as unrecorded. The others are marked as recorded.

/unrecorded/end

This consists of 2 extents of 1 block each. The last one is marked as unrecorded. The other is marked as recorded.

/unrecorded/beyond

This consists of 2 extents of 1 block each. The file shall have an information length of one block. The last one is marked as unrecorded.

References:

Applicable ECMA 167 references: 4/12, 4/14.14.1.1
Applicable UDF 2.0 references: NONE

Rules:

When accessing the files, reading bytes backed by unrecorded extents shall be seen as 0.

3.23 Information Length

Description:

Have files with and without a file tail. For files with a file tail, have one that begins on a block boundary and one that does not begin on a block boundary. Do the same for directories.

The information length of the file lg_eq_ad.txt will have the same value as the sum of the information lengths of its Allocation Descriptors.

The information length of the file lg_lt_ad.txt will be less than the sum of the information lengths of its Allocation Descriptors.

Both files shall contain non-zero value.

Discs:

All

Location on disc(s):

/Filelength

File names:

lg_eq_ad.txt
lg_lt_ad.txt

References:

Applicable ECMA 167 references: 4/14.9.10, 4/12.1
Applicable UDF 2.0 references: NONE

Rules:

All implementation shall be able to read the above files from start to logical length.
OSs that can distinguish between logical and physical length of a file shall report the correct physical length of both files.

3.24 Allocation Descriptor Test

Description:

Files shall be created in such a way that they use both short and long allocation descriptors.

Discs:

All

Location on disc(s):

/AD

Filenames:

/AD/sad/first_continuation
/AD/sad/last_continuation
/AD/sad/middle_continuation
/AD/sad/ads_with_zero_length
/AD/sad/file_with_all_and_rec_desc
/AD/sad/file_with_unall_and_unrec_desc
/AD/sad/file_with_last_unall_and_unrec_desc
/AD/sad/file_with_all_and_unrec_desc
/AD/sad/file_with_last_all_and_unrec_desc
/AD/sad/file_with_mult_cont_desc
/AD/lad/first_continuation
/AD/lad/last_continuation
/AD/lad/middle_continuation
/AD/lad/ads_with_zero_length
/AD/lad/file_with_all_and_rec_desc
/AD/lad/file_with_unall_and_unrec_desc
/AD/lad/file_with_last_unall_and_unrec_desc
/AD/lad/file_with_all_and_unrec_desc
/AD/lad/file_with_last_all_and_unrec_desc
/AD/lad/file_with_mult_cont_desc

Notes:

On discs 2-7, all files with short allocation descriptors shall have there file entries in the physical partition (UDF 2.3.10).

All the Ads shall be part of the file body.

File in directory /AD/sad will use short allocation descriptors and in directory /AD/lad will use long allocation descriptors.

Files in both /AD/sad and /AD/lad will have the same content.

Each of the files shall contain the offset repeated after every 8-bytes.

/AD/sad/first_continuation:
/AD/lad/first_continuation:

This file will have the first AD in the file entry as a continuation descriptor (see 14.14.1.1)

/AD/sad/last_continuation :
/AD/lad/last_continuation :
This file will have the file entry filled with ADs and the last AD is a continuation descriptor.

/AD/sad/middle_continuation :
/AD/lad/middle_continuation :
One of the middle ADs in the file entry will be a continuation entry.

/AD/sad/ads_with_zero_length :
/AD/lad/ads_with_zero_length :
The SAD file entry will terminate the allocation descriptor with a zero length AD. The total length is longer than the termination length to stress that the terminator is processed.
The LAD file entry will have zero length. The file size should match the sum of the lengths of the ADs before the AD with zero length, but the "Length of Allocation Descriptors" should cover the AD with zero length.

/AD/sad/file_with_all_and_rec_desc :
/AD/lad/file_with_all_and_rec_desc :
File will have ADs that are allocated and recorded.

/AD/sad/file_with_unall_and_unrec_desc :
/AD/lad/file_with_unall_and_unrec_desc :
File will have unallocated and unrecorded ADs.

/AD/sad/file_with_last_unall_and_unrec_desc :
/AD/lad/file_with_last_unall_and_unrec_desc :
Last AD in the file entry is unallocated and unrecorded.

/AD/sad/file_with_all_and_unrec_desc :
/AD/lad/file_with_all_and_unrec_desc :
All ADs will be allocated but unrecorded. The blocks covered by the ADs shall have non-zero data.

/AD/sad/file_with_last_all_and_unrec_desc :
/AD/lad/file_with_last_all_and_unrec_desc :
Last AD will be allocated but unrecorded. The block covered by this AD shall have non-zero data.

/AD/sad/file_with_mult_cont_desc :
/AD/lad/file_with_mult_cont_desc :
File entry will have more than one continuation extents.

References:

Applicable ECMA 167 references: 4/12, 4/14.14.1, 4/14.14.2

Rules:

Implementation must be able to read the files and compare the contents of the files with the same name in sad and lad directories.

3.25 Zero Length Files

Description:

Test the possible conditions of zero length files.

Discs:

All

Location on disc(s):

/Zero_len_files

Filenames:

>>Allocation descriptor space length is 0
zero_sz1
zero_sz2
zero_sz3

>>Allocation descriptor length is >16
zero_sz4
zero_sz5
zero_sz6

References:

Applicable ECMA 167 references: 4/12.1, 4/14.5 and 4/14.14
Applicable UDF 2.0 references: 2.1.3 (1.5, or 2.0)

Note: See ISO specification for an explanation of terms.

Rules:

Implementations shall meet the following requirements:

1/4=embbedded file
2/5=file specified by short ADs
3/6=file is specified by long ADs

In all cases the information length is 0.

Each file should be read accessible.

3.26 Extended Attributes (Mix)

Description:

Combine many of the EAs in several files, both embedded and in an external EA file, randomly including some FreeSpaceEAs.

Discs:

All

Location on disc(s):

/EA_mix

File names:

Name	Type	Comments
Many EAs, embedded	File	
Many EAs, external	File	
Mac Resource Fork, emb.	File	might not fit on 512 sector disk
Mac Resource Fork, ext.	File	

Mac Finder Info EA:

RBP	Content
0	Header Checksum
2	Reserved: #00 #00
4	Parent Dir ID: lower 32 bit of the Unique ID of the directory this file is stored in.
8	UDFFInfo: see below
24	UDFFXInfo: all #00 bytes
40	Resource Fork Data Length: 0
44	Resource Fork Allocated Length: 0

UDFFInfo:

RBP	Content
0	fdType: #72 #6F #74 #74 (reads 'ttro' in big endian)
4	fdCreator: #78 #74 #74 #74 (reads 'ttxt' in big endian)
8	fdFlags: #02 #00 (reads #0002 in big endian)
10	fdLocation: all #00s
14	fdFldr: #00 #00

Rules:

The first two files have one EA space each, with the same contents, once embedded, once in an EA file:

- File Times EA with Creation & Backup Date (UDF 1.5, 3.3.4.3)
- FreeEASpace (totalling 32 Bytes) (UDF 1.5, 3.3.4.5.1.1)
- DVD Copyright Information (UDF 1.5, 3.3.4.5.1.2)
- MacFinderInfo (contents: see above)
- Appl. Use EA (containing some informational text)

The "Mac Resource Fork ..." files have two EAs, of which one, the MacFinderInfo (specifying the length of the Macintosh Resource Fork), is always embedded, and the other, a valid Macintosh Resource Fork, is located in either EA space (embedded or external).

When embedded, the Resource Fork EA immediately follows the MacFinderInfo, when external, a FreeEASpace is put between the EA Header and the Resource Fork EA, filling up the first logical block, so that the Resource Fork EA starts on a block boundary. The Resource Fork EA size is rounded up to the logical block boundary.

On a Mac OS system, the files containing the FinderInfo EA shall show the SimpleText's Read-only icon (creator: 'ttxt', type: 'ttro') and be labeled with the first label (index 1), which is called "Essential" and colored orange on a english system by default.

3.27 Extended Attributes (Implementation Use)

Description:

Put an Implementation Use Extended Attribute into both a directory and a file to make sure that readers ignore it.

Discs:

All

Location on disc(s):

/EA_impuse

File names:

Name	Type	Comments
D_HasImpUseEA	Directory	
F_HasImpUseEA	File	Located inside the previous directory.

Rules:

Both File Entries have an embedded EA space that consists only of an empty Implementation Use EA.

The data of both file and directory is embedded in the FE.

3.28 Extended Attributes (non-embedded small EAs)

Description:

Have some of the smaller EAs (that are normally embedded) put into the non-embedded EA space.

Discs:

All

Location on disc(s):

/EA_small

File names:

Name	Type	Comments
Small EAs #1	File	Set A in FE, Set B in EA File
Small EAs #2	File	Set B in FE, Set A in EA File

Mac Finder Info EA:

RBP	Content
0	Header Checksum
2	Reserved: #00 #00
4	Parent Dir ID: lower 32 bit of the Unique ID of the directory this file is stored in.
8	UDFFInfo: see below
24	UDFFXInfo: all #00 bytes
40	Resource Fork Data Length: 0
44	Resource Fork Allocated Length: 0

UDFFInfo:

RBP	Content
0	fdType: #72 #6F #74 #74 (reads 'ttro' in big endian)
4	fdCreator: #78 #74 #74 #74 (reads 'ttxt' in big endian)
8	fdFlags: #02 #00 (reads #0002 in big endian)
10	fdLocation: all #00s
14	fdFldr: #00 #00

Rules:

There are two sets of small EAs in each of the files:

Set A:

- File Times EA with a creation date
- Appl. Use EA

Set B:

- Mac Finder Info EA (contents: see above)
- Appl. Use EA (different content from the one in Set A)

On a Mac OS system, the files containing the FinderInfo EA shall show the SimpleText's Read-only icon (creator: 'ttxt', type: 'ttro') and be labeled with the first label (index 1), which is called "Essential" and colored orange on a english system by default.

3.29 Extended Attributes (Mac Volume Info)

Description:

Add the Mac Volume Info EA to the root directorys EA space.

Discs:

All

Location on disc(s):

ICB of Root Directory

Rules:

In the File Entry of the Root directory, includes the MacVolumeInfo EA (UDF 2.0: 3.3.4.5.4.1) into the FEs EA space with the following fields in the EA:

Last Modification Date.
Last Backup Date.
Volume Finder Information: all zero bytes

The modification time for each file is:

Year:	1998
Month:	8
Day:	7
Hour:	6
Minute:	5
Second:	4
Centisecond:	3
Hundreds of microseconds:	2
Microseconds:	1

The backup time for each file is:

Year:	1998
Month:	8
Day:	8
Hour:	6
Minute:	5
Second:	4
Centisecond:	3
Hundreds of microseconds:	2
Microseconds:	1

3.30 Hard links

Description:

Have several hard links to a file.

Discs:

All

Location on disc(s):

/hardlink

File names:
/hardlink/hardln1

/hardlink/dir1/hardln2
/hardlink/dir1/hardln3
/hardlink/dir1/hardln4
/hardlink/dir1/hardln5

/hardlink/dir2/hardln6
/hardlink/dir2/hardln7

Note:

The files "hardln1", "hardln2" and "hardln6" are hard linked. Their content is "/hardlink/hardln1".

The files "hardln3" and "hardln4" are hard linked. Their content is "/hardlink/dir1/hardln3".

The files "hardln5" and "hardln7" are hard linked. Their content is "/hardlink/dir1/hardln5".

References:

ECMA 167 4/14.9.6

Rules:

For all operating systems:

Same file should have same content, length, timestamp and permissions.

For operating systems who support hard linked files, the link counter should be correct:

"hardln1" should have a link count of 3
"hardln3" should have a link count of 2
"hardln5" should have a link count of 2

3.31 Directory boundary conditions

Description:

Directories shall be created with size more than one logical block in size. And in each directory some parts of one file_id will cross logical block boundary.

Discs:

All (exception for disc 4)

Location on disc(s):

/directory_boundary

Filenames:

/directory_boundary/tag_crossing (Not on DISC4, Sequential writing 2.0)
/directory_boundary/iu_crossing
/directory_boundary/fid_crossing
/directory_boundary/tag_end_on_boundary
/directory_boundary/iu_end_on_boundary
/directory_boundary/fid_end_on_boundary

Notes:

All the directories shall be more than one logical block.
All logical blocks shall be recorded non-continuously.

tag_crossing - first word of the file_ids tag shall be in one logical block and the rest of the tag will be in the second logical block.

iu_crossing - file_id shall be recorded with non-zero implementation use length. Part of implementation use will be in one logical block and the rest will be in other logical block.

fid_crossing - Shall be recorded with part of the name of the file in one logical block and the rest will be in a different logical block.

tag_end_on_boundary – Shall be recorded with a tag ending precisely on the last byte of boundary and at least one more FID that follows

iu_end_on_boundary – Shall be recorded with an IU ending precisely on the last byte of boundary and at least one more FID that follows

fid_end_on_boundary – Shall be recorded with a FID ending precisely on the last byte of boundary and at least one more FID that follows

References:

Applicable ECMA 167 references: 4/14.4
Applicable UDF 2.0 references: 2.3.4, 2.3.4.4, and 2.3.4.5

Rules:

All implementations shall be able to read the above directories.

3.32 Volume name

Description:

Volume name: use Unicode characters with unusual characters (e.g., backspace, "\", "/", ":","[") in the first 11 characters of the volume name which has more than 27 characters. This is to test if a receiving system handles a volume name with Unicode and special characters correctly.

Location on disc(s):

D3 UDF 1.5 spared CD-RW

Volume Names:

All disks shall have the following volume names:

Disk Number	Volume Name
D1	OSTA_V1_D1
D2	OSTA_V1_D2
D3	/#0424h/:/[BS]\OSTA_V1_D3_TESTDISK
D4	OSTA_V1_D4
D5	OSTA_V1_D5
D6	OSTA_V1_D6
D7	OSTA_V1_D7

Rules:

If a receiving implementation does not support the characters used in the volume name, it should be able to translate the name into the OS supported characters and mount the volume.

Note:

All disks should have "*OSTA UDF Compliant" in the "DomainIdentifier" in the volume descriptor.

The value of 27 is the maximum volume label length under MacOS

3.33 Volume Recognition Sequence

Description:

An ISO 9660 descriptor is placed between the BEA01 and TEA01 descriptors
in the volume recognition sequence.

Discs:

D4, D7 (ISO 9660 bridge discs)

Location on Disc(s):

Volume Recognition Sequence starting with the first byte of the first sector that begins after byte
number 32767 of the volume recognition space.

References:

Applicable ECMA 167 references: 1/8.3
Applicable ISO 9660 references: 6.7, 8.1, 8.3, 8.4

Rules:

The volume recognition sequence for the bridge discs contains the ISO 9660 recognition
sequence (primary and supplementary volume descriptors followed by an ISO 9660 volume
descriptor set terminator). The block following the ISO 9660 terminator will contain a BEA01
descriptor followed by an NSR02 (D7) / NSR03 (D4) descriptor, a copy of the ISO9660 primary
volume descriptor, and the TEA01 terminating descriptor.

3.34 Tag serial number

Description:

Have FIDs with zero and matching the logical volume descriptor. Do for both files and directories.

Discs:

All

Location on disc(s):

/Tag_serial

Filenames:

zero
zero/file

nonzero
nonzero/file

Directory zero and the contained file shall be recorded with file entries whose descriptor tags record a tag serial number of zero.

Directory nonzero and the contained file shall be recorded with file entries whose descriptor tags record a tag serial number of #4321. The prevailing logical volume descriptor shall also be recorded with a descriptor tag recording a tag serial number of #4321. This serial number shall not be shared with any other logical volume descriptor recorded for the volume.

Both files shall be 64 bytes.

All other files and directories on the disc are unspecified with respect to this test.

References:

Applicable ECMA167 references: 4/7.2.5
Applicable UDF 2.0 references: 2.1.1

Rules:

All implementations shall be able to read the directories and files.

3.35 File Entry CRC

Description:

Have a descriptor with defective CRC, another one with bad checksum and then a valid descriptor.

Discs:

All

Location on disc(s):

/crc_errors

Filenames:

/crc_errors/bad_crc
/crc_errors/bad_cksm
/crc_errors/good

File bad_crc shall record a file entry whose descriptor tag records a descriptor CRC of #4321. This shall not be the valid CRC for the descriptor. All other values in the descriptor shall be valid.

File bad_cksm shall record a file entry whose descriptor tag records a checksum of #21. This shall not be the valid checksum for the descriptor tag. All other values in the descriptor shall be valid.

File good shall record a valid file entry.

All three files shall be 64 bytes.

References:

Applicable ECMA167 references: 4/7.2.6, 4/7.2.5
Applicable UDF 2.0 references: 2.2.1.2

Rules:

All implementations should fail to read "bad_crc".
All implementations should fail to read "bad_cksm".
All implementations shall read "good".

Note:

As mentioned in chapter 1, these test discs contain only correct and valid UDF with the exception of this case. Deliberately adding corrupt data however is the only way to ensure that a CRC failure will occur.

3.36 Volume Descriptor CRC

Description:

Have a descriptor with defective CRC, another one with bad checksum and then the valid descriptor. Have the bad ones point to an address outside the volume space.

Discs:

Disc 7

Location on disc(s):

The volume descriptor sequences shall be recorded in the following order

- A Logical Volume Descriptor (3:10.6) recording a descriptor tag with a CRC field containing #4321. This shall not be the valid CRC for the descriptor, and all other fields shall be valid. The partition map for this Logical Volume Descriptor shall contain one type 1 partition map (3:10.7.2) recording a partition number of #4321.
- A Partition Descriptor (3:10.5) recording a descriptor tag with a checksum field containing #4321. This shall not be the valid checksum for the descriptor tag, and all other fields shall be valid. The partition number for this descriptor shall be #4321. The partition starting location shall be #80000000 and the partition length shall be #1000.

Other specified and normal descriptors of the Volume Descriptor Sequence shall follow.

Filenames:

N/A

References:

Applicable ECMA167 references: 3/7.2.6, 3/8.4 and 3/7.2.3
Applicable UDF 2.0 references: NONE

Rules:

All implementations shall mount and otherwise operate on this volume normally.

Note:

As mentioned in chapter 1, these test discs contain only correct and valid UDF with the exception of this case. Deliberately adding corrupt data however is the only way to ensure that a CRC failure will occur.

3.37 Volume Descriptor Pointer

Description:

Split the necessary Volume Descriptor Sequence into 2 pieces so that you have to travel to both.

Discs:

Disc 7

Location on disc(s):

The Volume Descriptor Sequence shall be recorded in the following order

- Sector 0: The Logical Volume Descriptor.
- Sector 1: A Volume Descriptor Pointer referring to the allocation starting at the physical sector corresponding to Sector 16 in this specification.
- Sector 2-15: Terminating Descriptor (3:10.9). Note that although this sector contains a valid descriptor. It should be considered unrecorded space by any implementation, albeit not zeroes.
- Sectors 3-15: Sectors shall be filled with #00 bytes
- Sector 16: This begins the remainder of the Volume Descriptor Sequence, which shall be recorded as otherwise specified.

Filenames:

N/A

References:

Applicable ECMA167 references: 3/10.3
Applicable UDF 2.0 references: 2.2.3.1

Rules:

All implementations shall mount and otherwise operate on this volume normally.

3.38 Symbolic Links

Description:

Symbolic links shall be recorded referencing to a file and a directory and another Symbolic link.

Disc(s):

All

Location on disc(s):

/sym_link

Filenames:

/sym_link/dir   - This is a regular directory
/sym_link/dir/file   - This is a regular file
/sym_link/sym_link_to_dir   - The symbolic link shall reference "dir"
/sym_link/sym_link_to_file   - The symbolic link shall reference "file"
/sym_link/sym_link_to_sym_link   - This is a symbolic link referencing the previous
                                         symbolic link
/sym_link/larg_sym_link_to_file   - The symbolic link shall reference "file"

Notes:

/sym_link/sym_link_to_dir shall be a symbolic link to "dir".
/sym_link/sym_link_to_file shall be a symbolic link to "file".
/sym_link/sym_link_to_sym_link shall be a symbolic link to "sym_link_to_file".
/sym_link/larg_sym_link_to_file shall be a symbolic link to "file" but the
path name shall be longer than 512 bytes.

All symbolic links shall be made with relative path names as absolute path name can change
between operating systems and instances.

The large symbolic links shall make several hops as following "../sym_link/" for 42 times followed
by "file".

Rules:

Applicable ECMA 167 references: 4/14.6.6, 4:4.2.1.1
Applicable UDF 2.0 references: NONE

All operating system which choose to support symbolic links shall reference the file referenced
by the symbolic link and not the symbolic link for any "read" operations performed on the
symbolic link or when requesting attributes of the original file.

3.39 UID and GID

Description:

There will be several files with different UID and GID.

Disc(s):

All

Location on disc(s):

/u-gids

Filenames:

/u-gids/uid=00000000-gid=00000000
/u-gids/uid=00000FFF-gid=00000FFF
/u-gids/uid=30000000-gid=30000000
/u-gids/uid=FFFFFFFF-gid=FFFFFFFF

Notes:

All files will have the same value for uid and gid as mentioned in the file name.

If the operating system supports the concept of user/group IDs and does not support the above user/group IDs then it should use default user/group ID.

First three names are added to test operating systems with user/group IDs smaller than 32-bit. The last name is added to test the user/group id supporting operating system to work properly with media from an operating system that does not support user/group IDs.

3.40 Standard and Extended File Entries

Description:

Mix Standard and Extended File Entries in one directory to make sure that UDF 2.0 compliant File Systems handle both structures.

Discs:

All discs with UDF 2.0 structures: D4, D5

Location on disc(s):

/ExtdFEs

File names:

Name	Type	Comments
1of5	File	Standard File Entry
2of5	File	*Extended* File Entry
3of5	Directory	Standard File Entry
3of5/file_in_3of5	File	Standard File Entry
4of5	Directory	*Extended* File Entry
4of5/file_in_4of5	File	Standard File Entry
5of5	File	Standard File Entry

Rules:

The main directory contains 3 files and 2 directories. Each of the two directories contains a simple file identifying its parent directory.

A UDF 2.0 compliant File System shall be able to access all items without errors.

4 File Content Testing

File Content

All files, unless otherwise specified, shall contain the following unique verifiable data.
Each 512-byte block of a file will be structured as follows:

struct FileRecord {
Uint32 RecordNumber; /* Given 512 byte records, this is the record number (1-n) */
Uint8 LengthOfName; /* Length of the NameOnDisk field */
Uint8 NameOnDisk[]; /* The original file name as recorded on disc */
/* The record shall be padded with 0 to fill out the 512 bytes */
}

Uint32 as defined in 1/7.1.5 in ECMA167

This type of file contents is intended to help a developer when they encounter a problem
reading the file data. The actual file size will be dependent upon the purpose of a file and will be
specified by the corresponding test associated with the file.

File CRC

Part of the verification test process will be to verify that an implementation can correctly
read the contents of every file on the test disc. This is accomplished by the calculation of
a 16-bit CRC on the file content using the algorithm defined in UDF. For each supported
operating system (OS) there shall exist an OS specific subdirectory in the root directory of each
test disc. The following file shall exist in each OS specific subdirectory.

FILECRC.TXT

This file shall contain a list of all files located on the test
disc, as well as the CRC of each file. The fully qualified file names listed in
this file shall be the file names as they appear to the user, after going through
the OSTA name translation algorithm for the corresponding OS.
The following is an example of what the FILECRC.TXT might contain for the Windows
9X operating system.

"/README.TXT", 0x7034
"/Dates/DateFile1.txt", 0x0A39
"/AllocationDescriptors/Short/TestFile1", 0x45F3

Note:

The filename shall include the pathname for the file.
For every file on the test disc an implementation shall calculate the file its 32-bit CRC and
compare it against the master CRC in the FILECRC.TXT file. If the CRC does not
match, the implementation has failed to read the file successfully.