Standard ECMA-167
2nd Edition - December 1994

ECMA

Standardizing Information and Communication Systems

Volume and File Structure for Write-Once and Rewritable Media using Non-Sequential Recording for Information Interchange

Phone: +41 22 849.60.00 - **Fax:** +41 22 849.60.01 - **URL:** http://www.ecma.ch - **Internet:** helpdesk@ecma.ch

Standard ECMA-167
2nd Edition - December 1994

ECMA

Standardizing Information and Communication Systems

Volume and File Structure for Write-Once and Rewritable Media using Non-Sequential Recording for Information Interchange

Part 1 - General

Part 2 - Volume and boot block recognition

Part 3 - Volume structure

Part 4 - File structure

Part 5 - Record structure

Phone: +41 22 849.60.00 - **Fax:** +41 22 849.60.01 - **URL:** http://www.ecma.ch - **Internet:** helpdesk@ecma.ch

MB- Ecma-167.doc - 06.09.96 12:04

Brief History

This ECMA Standard is a volume and file structure standard for interchanging files and as such, it is a peer to existing volume and file structure standards such as ECMA-107 and ECMA-119 . It is rather different from those standards in at least two important ways. Firstly, it offers much more functionality, mainly because of user needs for increased character set support and for more powerful file system features. Secondly, it acknowledges the separate concerns of booting, volume structure and file system structure. Rather than bundling these different functions together, this ECMA Standard carefully segregates these functions into separate parts and describes in detail how those parts fit together. It is expected that future volume and file structure standards will fit into this framework, rather than building other distinct and incompatible formats.

This ECMA Standard consists of five Parts published in one Volume. Part 1 - General - specifies references, definitions, notations and basic structures used in the other four Parts. Part 2 - Volume and boot block recognition - specifies formats and system requirements for recognising the volume structures on a medium and booting from a medium. Part 3 - Volume structure - specifies how to record various volume-related entities such as volumes, volume set s and logical volumes. Part 4 - File structure - specifies how to record and interpret files, both file data and file attributes, and file hierarchies within logical volumes. Part 5 - Record structure - specifies how to record and interpret file data encoded as records.

This ECMA Standard has been adopted by the ECMA General Assembly of December 1994.

Table of Contents

Standard ECMA - 167

Volume and File Structure of Write-Once and Rewritable Media using Non-Sequential Recording for Information Interchange

Part 1: General

1 Scope

This ECMA Standard specifies a format and associated system requirements for volume and boot block recognition, volume structure, file structure and record structure for the interchange of information on media between users of information processing systems.

The media shall be recorded as if the recording of sectors may be done in any order.

Note 1

The medium is not restricted to being of only one type; the type of medium may be either write once, or read only, or rewritable, or a combination of these types.

This ECMA Standard consists of the following five Parts:

Part 1: General

Part 2: Volume and Boot Block Recognition

Part 3: Volume Structure

Part 4: File Structure

Part 5: Record Structure

Annex A - ICB Strategies, is part of Part 4.

Part 1 specifies references, definitions, notation and basic structures that apply to the other four Parts.

2 Parts references

The first digit of a reference within this ECMA Standard identifies the Part, e.g. 2/5 refers to clause 5 in Part 2, and figure 4/3 refers to figure 3 in Part 4.

3 Conformance

3.1 Conformance of a medium

A medium shall be in conformance with this ECMA Standard when it conforms to a standard for recording (see 1/5.10) and information recorded on sectors of the medium conform to the specifications of Part 1 and one or more of Parts 2, 3, 4 and 5. A statement of conformance shall identify the sectors of the medium on which information is recorded according to the specifications of this ECMA Standard, and the Parts and the levels of medium interchange (see 2/10, 3/11, and 4/15) to which the contents of those sectors of the medium conform.

3.2 Conformance of an information processing system

An information processing system shall be in conformance with this ECMA Standard if it meets the requirements specified in Part 1 and one or more of Parts 2, 3, 4 and 5 either for an originating system (see 2/12, 3/13, 4/17 and 5/11) or for a receiving system (see 2/13, 3/14, 4/18 and 5/12) or for both types of system. A statement of conformance shall identify the Parts, and the levels of the requirements for each of those Parts, which can be met by the system.

4 References

ECMA-6	7-Bit Coded Character Set (1991)
ECMA-35	Code Extension Techniques (1994)
ECMA-48	Control Functions for Coded Character Sets (1991)
ECMA-94	8-Bit Single-Byte Coded Graphic Character Sets - Latin Alphabets No. 1 to No. 4 (1986)
ECMA-107	Volume and File Structure of Flexible Disk Cartridges for Information Interchange (1995)
ECMA-119	Volume and File Structure of CDROM for Information Interchange (1987)

ECMA-168 Volume and File Structure for Read-Only and Write-Once Compact Disk Media for Information Interchange (1994).

5 Definitions

For the purpose of this ECMA Standard, the following definitions apply.

5.1 application

A program that processes the contents of a file, and may also process selected attribute data relating to the file or to the volume(s) on which the file is recorded.

5.2 byte

A string of eight binary digits operated upon as a unit. If the standard for recording (see 1/5.10) specifies that the container for the recording of a byte has more than eight bits, in this ECMA Standard a byte shall be recorded in the least significant eight bits of the container with the remaining bits of the container set to ZERO.

5.3 descriptor

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

5.4 file

A collection of information.

5.5 implementation

A set of processes which enable an information processing system to behave as an originating system , or as a receiving system, or as both types of system.

5.6 originating system

An information processing system which can create a set of files on a volume set for the purpose of data interchange with another system.

5.7 receiving system

An information processing system which can read a set of files from a volume set which has been created by another system for the purpose of data interchange

5.8 record

A sequence of bytes treated as a unit of information.

5.9 sector

The data field of the smallest addressable part of the medium that can be accessed independently of other addressable parts of the medium.

5.10 standard for recording

A standard that specifies the recording method and the addressing method for the information recorded on a medium. The specifications of the standard for recording that are relevant for this ECMA Standard are:

- a unique address for each sector;
- the length of each sector;
- the means for determining whether a sectoris read-only, write-once, or rewritable;
- for media where sectors may only be recorded once, a means for detecting whether each sector has not yet been recorded;
- whether sectors may require preprocessing prior to recording.

The standard for recording used in conjunction with this ECMA Standard is subject to agreement between the originator and recipient of the medium.

5.11 user

A person or other entity (for example, an application) that causes the invocation of the services provided by an implementation.

5.12 volume

A sector address space as specified in the relevant standard for recording.

Note 2

A medium usually has a single set of sector addresses, and is therefore a single volume. A medium may have a separate set of addresses for each side of the medium, and is therefore two volumes.

5.13 volume set

A collection of one or more volumes with identical volume set identification.

6 Notation

The following notation is used in this ECMA Standard:

6.1 Numerical notation

6.1.1 Decimal notation

Numbers in decimal notation are represented by decimal digits.

6.1.2 Hexadecimal notation

Numbers in hexadecimal notation are represented as a sequence of one or more hexadecimal digits prefixed by "#":

hexadecimal digit	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
decimal value	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

6.2 Bit fields

Certain fields containing an integral value, or parts of fields containing an integral value, are intended to be interpreted as an array of bits. This array of bits shall be referred to as a bit field

Bit positions within an *n* bit field are numbered such that the least significant bit is numbered 0 and the most significant bit is numbered *n* - 1.

6.3 Descriptor formats

Descriptor formats shall be specified by a figure specifying the location, length, name and contents of each field. The interpretation of each field shall be given in the prose associated with the figure.

Byte position	Length in bytes	Name	Contents
0	4	Data Length (=D_L)	`Uint32` (1/7.1.5)
4	32	Application Identifier	`regid` (1/7.4)
36	4	Reserved	#00 bytes
40	2	Type	`Int16` (7.1.4) =57
42	D_L	Implementation Use	bytes
[D_L+42]	*	Padding	#00 bytes

Figure 1 - Example descriptor format

The descriptor specified by figure1/1 has six fields:

– The Data Length field shall be a 32-bit unsigned integer recorded according to 1/7.1.5 in byte positions 0 to 3 of the descriptor. The value of this field may be referred to as D_L.

- The Application Identifier field shall be a 32 byte field specifying an identification of an application recorded according to 1/7.4 in byte positions 4 to 35 of the descriptor.
- The Reserved field shall be 4 bytes, each with the value #00, recorded in byte positions 36 to 39 of the descriptor.
- The Type field shall be the number 57 as a 16-bit signed integer recorded according to 1/7.1.4 in byte positions 40 to 41 of the descriptor.
- The Implementation Use field shall be D_L bytes recorded in byte positions 42 to (D_L+41), where D_L is the value recorded in the Data Length field of this descriptor. A symbolic length referred to in a descriptor shall either be defined within that descriptor or be described in the interpretation of the field it is used in. The specification of the interpretation for this field might state that the interpretation of those bytes is not specified by this ECMA Standard, or could specify some specific interpretation for those bytes.
- The Padding field shall be a variable length field, as indicated by the asterisk "*", of bytes, each with a value of #00. The specification of the interpretation for the field shall specify the length of the field.

6.4 Character strings

A value for a sequence of bytes may be specified by a quoted sequence of characters, encoded according to the International Reference Version of ECMA-6 . For example, "Sheep" shall represent the bytes #53, #68, #65, #65, #70.

6.5 Arithmetic notation

The notation *ip(x)* shall mean the integer part of *x*.

The notation *rem(a,b)* shall mean $a - b \times ip(a/b)$, where *a* and *b* are integers.

6.6 Schema

The notation specified by this clause, hereafter referred to as schema , specifies the format of a structure, or sequence of structures, by construction. White space is unimportant. A structure shall be a sequence of terms. A term shall be either a name enclosed by `<>` or a structure definition enclosed by `{}`. A term may be given a name *label* by preceding the term with `[ label ]`. A term may be suffixed by one of the repetition operators in figure 1/2.

Operator	Interpretation
n + *m*	*n* to *m* occurrences inclusive
n+	*n* or more occurrences
n	*n* occurrences exactly

Figure 2 - Repetition operators

The expression *term1* | *term2* means either *term1* or *term2* shall appear at this place in the sequence.

Names shall be resolved in one of the following three ways:

- the name is that of a descriptor or term defined in this ECMA Standard
- the name has been defined in this structure definition using the [ ] notation
- the name will be defined in the prose associated with the structure definition

If a term is followed by a clause enclosed in ( ), it shall refer to only those objects specified by the term for which the clause is true.

These operators shall be applied in increasing order of precedence with the| operator having lowest precedence:

| repetition operator [ ] ( )

As an example, the schema shown in figure 1/3, specifies that the term Set means zero or more Groups, where a Group is a sequence of two or more Group Headers, followed by a Group Element, which is one of three alternatives (one or two Type-1 Descriptors, or a single Type-2 Descriptor whose length is even, or one or more Type-3 Descriptors), followed by one or more Group Trailers.

```
[Set]{
        [Group]{
              <Group Header> 2+
              [Group Element]{
                    <Type-1 Descriptor> 1+2
                  |  <Type-2 Descriptor>(descriptor length is even)
                  |  <Type-3 Descriptor> 1+
              }
              <Group Trailer> 1+
        } 0+
}
```

Figure 3 - Example schema

6.7 Other notations

Various other notations used in this ECMA Standard are specified in figure 1/4.

Notation	Interpretation
BP	Byte position within a descriptor, starting with 0
RBP	Relative byte position within a descriptor, starting with 0
ZERO	A single bit with the value 0
ONE	A single bit with the value 1

Figure 4 - Other notations

7 Basic types

The following basic types are used in this ECMA Standard.

7.1 Numerical values

The recording format of a numerical value represented in binary notation by an *n*-bit number shall be denoted by a type name of `Int`*n* or `Uint`*n* where

- *n* denotes the number of bits used in the binary number
- Uint denotes an unsigned integer x, in the range $0 \leq x < 2n$, represented as a binary number
- Int denotes a signed integer *x*, in the range $-2^{n-1} < x < 2^{n-1}$, represented by a two's complement number

A numerical value shall be recorded in a field of a structure specified by this ECMA Standard in one of the following formats. The applicable format shall be specified in the description of the structure.

7.1.1 8-bit unsigned numerical values

A `Uint8` value shall be recorded as an 8-bit unsigned number in a one-byte field.

7.1.2 8-bit signed numerical values

An `Int8` value shall be recorded as a two's complement number in a one-byte field.

7.1.3 16-bit unsigned numerical values

A `Uint16` value, represented by the hexadecimal representation #wxyz, shall be recorded in a two-byte field as #yz #wx.

Note 3

For example, the decimal number 4 660 has #1234 as its hexadecimal representation and shall be recorded as #34 #12.

7.1.4 16-bit signed numerical values

An `Int16` value, represented in two's complement form by the hexadecimal representation #wxyz, shall be recorded in a two-byte field as #yz #wx.

Note 4

For example, the decimal number -30 875 has #8765 as its hexadecimal representation and shall be recorded as #65 #87.

7.1.5 32-bit unsigned numerical values

A `Uint32` value, represented by the hexadecimal representation #stuvwxyz, shall be recorded in a four-byte field as #yz #wx #uv #st.

Note 5

For example, the decimal number 305 419 896 has #12345678 as its hexadecimal representation and shall be recorded as #78 #56 #34 #12.

7.1.6 32-bit signed numerical values

An `Int32` value, represented in two's complement form by the hexadecimal representation #stuvwxyz, shall be recorded in a four-byte field as #yz #wx #uv #st.

Note 6

For example, the decimal number -559 038 737 has #DEADBEEF as its hexadecimal representation and shall be recorded as #EF #BE #AD #DE.

7.1.7 64-bit unsigned numerical values

A `Uint64` value, represented by the hexadecimal representation #klmnopqrstuvwxyz, shall be recorded in an eight-byte field as #yz #wx #uv #st #qr #op #mn #kl.

Note 7

For example, the decimal number 12 345 678 987 654 321 012 has #AB54A9A10A23D374 as its hexadecimal representation and shall be recorded as #74 #D3 #23 #0A #A1 #A9 #54 #AB.

7.2 Character sets and coding

Except as specified in this clause, the characters in the descriptors specified by this ECMA Standard shall be coded according to ECMA-6

Certain fields specifying character strings shall be designated as containing either a `dstring` (1/7.2.12) or d-characters. The specification of the d-characters allowed in these fields and the method of recording shall be specified by a`charspec`, defined in 1/7.2.1. The set of allowed characters shall be referred to as d-characters.

Note 8

Support for a variety of character sets is a requirement for this ECMA Standard. Ideally, there would be only one character standard used. In practice, several standards, including ECMA-6, ECMA-35, ECMA-94, Latin Alphabet No. 1 and ISO/IEC 10646-1 are used. This ECMA Standard accommodates current practice by specifying several character sets and providing a mechanism for specifying other character sets.

As an example, CS2 (see 1/7.2.4) uses ECMA-6 as the base character set but restricts fields containing characters to a widely usable subset of this character set.

7.2.1 Character set specification

The set of characters allowed in certain descriptor fields shall be specified by a `charspec`, which shall be recorded in the format shown in figure 1/5.

RBP	Length	Name	Contents
0	1	Character Set Type	`Uint8` (1/7.1.1)
1	63	Character Set Information	bytes

Figure 5 - `charspec` format

7.2.1.1 Character Set Type (RBP 0)

This field shall specify the allowed characters by identifying a set of characters shown in figure 1/6.

Type	Allowed characters
0	The CS0 coded character set (1/7.2.2).
1	The CS1 coded character set (1/7.2.3).
2	The CS2 coded character set (1/7.2.4).
3	The CS3 coded character set (1/7.2.5).
4	The CS4 coded character set (1/7.2.6).
5	The CS5 coded character set (1/7.2.7).
6	The CS6 coded character set (1/7.2.8).
7	The CS7 coded character set (1/7.2.9).
8	The CS8 coded character set (1/7.2.10).
9-255	Reserved for future standardisation.

Figure 6 - Sets of allowed characters

Note 9

Briefly, these character sets are:

CS0 ¾ by agreement

CS1 ¾ the whole or any subset of the graphic characters specified by ECMA-6

CS2 ¾ a highly portable set of 38 graphic characters which include the characters in ECMA-119 file identifiers associated with a directory hierarchy identified by an ECMA-119 Primary Volume Descriptor

CS3 ¾ the 63 graphic characters of the portable ISO/IEC 9945-1 file name set

CS4 ¾ the 95 graphic characters of the International Reference Version of ECMA-6

CS5 ¾ the 191 graphic characters of ECMA-94, Latin Alphabet No. 1

CS6 ¾ a set of graphic characters that may be identified by ECMA-35 and ECMA-48

CS7 ¾ a set of graphic characters that may be identified by ECMA-35 and ECMA-48 and, optionally, code extension characters using ECMA-35 and ECMA-48

CS8 ¾ a set of 53 graphic characters that are highly portable to most personal computers

7.2.1.2 Character Set Information (RBP 1)

Except where specified in the following specifications of character sets CS0 through CS8, the contents of this field shall be set to all #00 bytes.

Note 10

The Character Set Types CS0, CS1, CS6 and CS7 require the use of the Character Set Information field to specify a set of graphic characters. CS1 restricts the set of graphic characters to those specified by ISO/IEC 10646-1. CS0, CS6 and C7 are not restricted to any particular set of graphic characters. CS7 allows code extension characters (see 1/7.2.9.1) to be used in a descriptor field. The same set of graphic characters may be specified by using the CS0, CS1, CS6 or CS7 Character Set Types. The order of specifying the escape sequences and control sequences in a Character Set Information field is not specified by this ECMA Standard. For example, in specifying a character set, the escape sequence identifying the G1 character set may be recorded before the escape sequence specifying the G0 character set. Character Set Information fields with different byte sequences may actually be identifying the same set of graphic characters.

7.2.2 CS0 character set

The CS0 character set and its d-characters shall be subject to agreement between the originator and recipient of the medium.

An identification of the character set may be given in the Character Set Information field. Such identification shall be recorded contiguously from the start of the field and any unused bytes shall be set to #00.

7.2.3 CS1 character set

The CS1 d-characters shall be the graphic characters of the character sets specified by the Character Set Information field.

The Character Set Information field shall specify one or more escape sequences, control sequences or both escape sequences and control sequences to be used in an 8-bit environment according to ECMA-35 and ECMA-48 that designate and implicitly invoke graphic character sets specified in ISO/IEC 10646-1. These sequences shall be recorded contiguously from the start of the field and any unused bytes shall be set to #00.

7.2.4 CS2 character set

The CS2 d-characters shall be the 38 characters in positions 02/14, 03/00 to 03/09, 04/01 to 05/10, and 05/15 of the International Reference Version of ECMA-6. The Character Set Information field shall be set to all #00 bytes.

Note 11

These characters are: FULL STOP, DIGITs, LATIN CAPITAL LETTERs and LOW LINE.

7.2.5 CS3 character set

The CS3 d-characters shall be the 65 characters in positions 02/13 to 02/14, 03/00 to 03/09, 04/01 to 05/10, 05/15, and 06/01 to 07/10 of the International Reference Version of ECMA-6. The Character Set Information field shall be set to all #00 bytes.

Note 12

These characters are: HYPHEN-MINUS, FULL STOP, DIGITs, LATIN CAPITAL LETTERs, LATIN SMALL LETTERs and LOW LINE.

7.2.6 CS4 character set

The CS4 d-characters shall be the 95 characters in positions 02/00 to 07/14 of the International Reference Version of ECMA-6 The Character Set Information field shall be set to all #00 bytes.

7.2.7 CS5 character set

The CS5 d-characters shall be the 191 characters in positions 02/00 to 07/14 and 10/00 to 15/15 of ECMA-94, Latin Alphabet No. 1. The Character Set Information field shall be set to all #00 bytes.

7.2.8 CS6 character set

The CS6 d-characters shall be the graphic characters of the character sets specified by the Character Set Information field.

The Character Set Information field shall specify one or more escape sequences, control sequences or both escape sequences and control sequences according to ECMA-35 and ECMA-48 that designate and implicitly invoke the graphic character sets to be used in an 8-bit environment according to ECMA-35 or ISO/IEC 10646-1. These sequences shall be recorded contiguously from the start of the field and any unused bytes shall be set to #00.

7.2.9 CS7 character set

The CS7 d-characters shall be the graphic characters of the character sets specified by the Character Set Information field and code extension characters (see 1/7.2.9.1).

The Character Set Information field shall specify one or more escape sequences, control sequences or both escape sequences and control sequences according to ECMA-35 and ECMA-48 that designate and implicitly invoke the graphic character sets to be used in an 8-bit environment according to ECMA-35 or ISO/IEC 10646-1. These sequences shall be recorded contiguously from the start of the field and any unused bytes shall be set to #00.

7.2.9.1 Code extension characters

A descriptor field which has been assigned to contain d-characters specified by the CS7 Character Set may include one or more of the following, referred to as code extension characters, to allow alternative character sets to be recorded in the descriptor field.

- Escape sequences according to ECMA-35 or ISO/IEC 10646-1.
- Shift functions according to ECMA-35
- Control functions according to ECMA-48 or ISO/IEC 10646-1.

7.2.10 CS8 character set

The CS8 d-characters shall be the 53 characters in positions 02/01, 02/03 to 02/09, 02/13 to 02/14, 03/00 to 03/09, 04/00 to 05/10, 05/14 to 06/00, 07/11 and 07/13 to 07/14 of the International Reference Version of ECMA-6.

Note 13

These characters are: EXCLAMATION MARK, NUMBER SIGN, DOLLAR SIGN, PERCENT SIGN, AMPERSAND, APOSTROPHE, LEFT PARENTHESIS, RIGHT PARENTHESIS, HYPHEN-MINUS, FULL STOP, DIGITs, LATIN CAPITAL LETTERs, CIRCUMFLEX ACCENT, LOW LINE, GRAVE ACCENT, LEFT CURLY BRACKET, RIGHT CURLY BRACKET, TILDE.

7.2.11 List of character sets

A list of Character Set Types (see 1/7.2.1.1) shall be recorded as a `Uint32` (1/7.1.5) where the bit for a Character Set Type shall be ONE if that Character Set Type belongs to the list and ZERO otherwise.

The bit for Character Set Type CS*n* shall be recorded in bit *n* of the `Uint32` (1/7.1.5). Bits 9-31 are reserved for future standardisation and shall be set to ZERO.

7.2.12 Fixed-length character fields

A `dstring` of length *n* is a field of *n* bytes where d-characters (1/7.2) are recorded. The number of bytes used to record the characters shall be recorded as a `Uint8` (1/7.1.1) in byte *n*, where *n* is the length of the field. The characters shall be recorded starting with the first byte of the field, and any remaining byte positions after the characters up until byte *n-1* inclusive shall be set to #00.

Unless otherwise specified, a `dstring` shall not be all #00 bytes.

7.3 Timestamp

A `timestamp` shall specify a date and time recorded in the format shown in figure 1/7. If all fields are 0, it shall mean that the date and time are not specified.

RBP	Length	Name	Contents
0	2	Type and Time Zone	`Uint16` (1/7.1.3)
2	2	Year	`Int16` (1/7.1.4)
4	1	Month	`Uint8` (1/7.1.1)
5	1	Day	`Uint8` (1/7.1.1)
6	1	Hour	`Uint8` (1/7.1.1)
7	1	Minute	`Uint8` (1/7.1.1)
8	1	Second	`Uint8` (1/7.1.1)
9	1	Centiseconds	`Uint8` (1/7.1.1)
10	1	Hundreds of Microseconds	`Uint8` (1/7.1.1)
11	1	Microseconds	`Uint8` (1/7.1.1)

Figure 7 - `timestamp` format

7.3.1 Type and Time Zone (RBP 0)

The most significant 4 bits of this field, interpreted as a 4-bit number, shall specify the interpretation of the `timestamp` as shown in figure 1/8. The least significant 12 bits, interpreted as a signed 12-bit number in two's complement form, shall be interpreted as follows:

– If the value is in the range −1 440 to 1 440 inclusive, then the value specifies the offset, in minutes, of the date and time of the day from Coordinated Universal Time

– If the value is −2 047, then no such value is specified.

Type	Interpretation
0	The `timestamp` specifies Coordinated Universal Time
1	The `timestamp` specifies local time
2	The interpretation of the `timestamp` is subject to agreement between the originator and recipient of the medium.
3-15	Reserved for future standardisation.

Figure 8 - `timestamp` interpretation

7.3.2 Year (RBP 2)

This field shall specify the year as a number in the range 1 to 9999.

7.3.3 Month (RBP 4)

This field shall specify the month of the year as a number in the range 1 to 12.

7.3.4 Day (RBP 5)

This field shall specify the day of the month as a number in the range 1 to 31.

7.3.5 Hour (RBP 6)

This field shall specify the hour of the day as a number in the range 0 to 23.

7.3.6 Minute (RBP 7)

This field shall specify the minute of the hour as a number in the range 0 to 59.

7.3.7 Second (RBP 8)

If the value of the Type field is 2, then this field shall specify the second of the minute as a number in the range 0 to 60. Otherwise, this field shall specify the second of the minute as a number in the range 0 to 59.

7.3.8 Centiseconds (RBP 9)

This field shall specify the hundredths of the second as a number in the range 0 to 99.

7.3.9 Hundreds of Microseconds (RBP 10)

This field shall specify the hundreds of microseconds as a number in the range 0 to 99.

7.3.10 Microseconds (RBP 11)

This field shall specify the microseconds as a number in the range 0 to 99.

7.4 Entity identifier

A `regid` specifies an entity identification and shall be recorded in the format shown in figure 1/9. The identification in a `regid` pertains to certain information; this information shall be called the scope of the `regid`. The scope of a `regid` consists of the field in which the `regid` is recorded and any information specified by the description of that field to be part of the scope the regid.

RBP	Length	Name	Contents
0	1	Flags	`Uint8` (1/7.1.1)
1	23	Identifier	bytes
24	8	Identifier Suffix	bytes

Figure 9 - `regid` format

7.4.1 Flags (RBP 0)

This field shall specify certain characteristics of the `regid` as shown in figure 1/10.

Bit	Interpretation
0	Dirty: If an implementation modifies the information on the medium within the scope of this `regid` such that the identification specified by this `regid` might not be valid, then this bit shall be set to ONE, otherwise it shall be set to ZERO.
1	Protected: If this bit is ONE, then the contents of this `regid` shall not be modified; if this bit is ZERO, then the contents of this `regid` may be modified (see 3/13.1 and 4/17.2.3).
2-7	shall be reserved for future standardisation and all bits shall be set to ZERO.

Figure 10 - Characteristics of `regid`

7.4.2 Identifier (RBP 1)

If the first byte of this field contains #2B, then this field contains an identifier specified by ECMA-168 or this ECMA Standard. If the first byte of this field contains #2D, then this field contains an identifier that shall not be registered. If the first byte of this field is neither #2D nor #2B, then this field shall specify an identifier which may be registered according to ISO/IEC 13800. An identifier shall be a sequence of at most 23 octets, at least one of which shall be nonzero; these octets shall be recorded in ascending order as the least significant 8 bits of bytes 0 through 22 of this field respectively. Any unused bytes shall be set to #00.

The interpretation of the content of the Identifier field shall be specified in the description of the descriptor field in which the `regid` is recorded.

If this field contains all #00 bytes, then this field does not specify an identifier.

Note 14

The values #2B and #2D do not represent characters. However, for most coded character sets using the value recorded in one byte to represent a character, such as ECMA-6, the value #2B corresponds to "+" and the value #2D corresponds to "-".

7.4.3 Identifier Suffix (RBP 24)

This field shall specify further identification in a manner not specified by this ECMA Standard.

Standard ECMA - 167

Volume and File Structure of Write-Once and Rewritable Media using Non-Sequential Recording for Information Interchange

Part 2 : Volume and Boot Block Recognition

Section 1 - General

1 Scope

Part 2 specifies a format and associated system requirements for volume and boot block recognition by specifying:

- volume recognition;
- boot descriptors intended for use to bring a system to a known state;
- levels of medium interchange;
- requirements for the processes which are provided within information processing systems, to enable information to be interchanged between different systems; for this purpose, Part 2 specifies the functions to be provided within systems which are intended to originate or receive media which conform to Part 2.

2 Parts references

See 1/2.

3 Part interface

This clause specifies the interface of Part 2 to other standards or Parts.

3.1 Input

Part 2 requires the specification of the following by another standard or Part.

- A standard for recording (see 1/5.10).
- The address of the initial sector in the volume (see 2/8.1.1).
- A volume recognition space (see 2/8.2).

3.2 Output

Part 2 specifies the following which may be used by other standards or Parts.

- identification of certain standards (see 2/9.1.2) used to record information in the volume.
- information that may be used to bring a system to a known state.

4 Conformance

See 1/3.

5 Definitions

In addition to the definitions of Part 1 (see 1/5), the following definition applies for Part 2.

5.1 extent

A set of sectors, the sector numbers of which form a continuous ascending sequence. The address, or location, of an extent is the number of the first sector in the sequence.

6 Notation

The notation of Part 1 (see 1/6) applies to Part 2.

7 Basic types

The basic types of Part 1 (see 1/7) apply to Part 2.

Section 2 - Requirements for the medium for volume and boot block recognition

8 Volume recognition

8.1 Arrangement of data on a volume

8.1.1 Sector numbers

Each sector of a volume shall be identified by a unique sector number. Sector numbers shall be consecutive integers assigned in an ascending sequence, in the order of ascending physical address of the volume as specified in the relevant standard for recording (see 1/5.10). Sector number 0 shall be assigned to the initial sector of the volume as specified in 2/3.1.

8.2 Volume recognition space

A volume recognition space shall be a contiguous sequence of sectors. The bytes in the volume recognition space shall be numbered with consecutive integers assigned in ascending sequence. The numbering shall start from 0 which shall be assigned to the first byte of the first sector of the volume recognition space. The numbering shall continue through successive bytes of that sector and then through successive bytes of each successive sector, if any, of the volume recognition space.

8.3 Volume recognition area

A volume recognition area shall be recorded in the volume recognition space. A volume recognition area shall consist of a volume recognition sequence (see 2/8.3.1) recorded in consecutively numbered sectors starting with the first byte of the first sector that begins after byte number 32 767 of the volume recognition space. Part 2 does not specify the interpretation of the information recorded in the volume recognition space other than in the volume recognition area of the volume recognition space.

8.3.1 Volume recognition sequence

A volume recognition sequence shall consist of a consecutively recorded sequence of one or more Volume Structure Descriptors (see 2/9.1) recorded according to the schema shown in figure 2/1.

Each Volume Structure Descriptor shall specify a standard or clause which shall specify the interpretation of the contents of the descriptor and the value of *n* (see figure 2/1).

The first Volume Structure Descriptor of the sequence shall be recorded beginning at the first byte of the first sector of the volume recognition area in which it is recorded. Each successive Volume Structure Descriptor of the sequence shall be recorded beginning at the first byte of the sector with the next higher sector number than that of the last sector constituting the previous Volume Structure Descriptor of the sequence.

Note 1

The volume recognition sequence is terminated by the first sector which is not a valid descriptor, rather than by an explicit descriptor. This sector might be an unrecorded or blank sector

```
[volume recognition sequence]{
   <CD-ROM Volume Descriptor Set>0+1
   [Extended Area]{
         <Beginning Extended Area Descriptor> 1+
         { <Volume Structure Descriptor> | <Boot Descriptor> } n+
         <Terminating Extended Area Descriptor> 1+
   } 0+
}
```

Figure 1 - Volume recognition sequence schema

8.3.1.1 CD-ROM Volume Descriptor Set

A CD-ROM Volume Descriptor Set shall be a set of consecutively recorded Volume Structure Descriptors whose Standard Identifier fields shall not contain “BEA01” and shall be interpreted according to ECMA-119.

8.4 Recording of descriptors

All the descriptors in Part 2 shall be recorded so that the first byte of the descriptor coincides with the first byte of a sector. All space, if any, after the last byte of the descriptor up to the end of the sector containing the last byte of the descriptor is reserved for future standardisation and shall be recorded as all #00 bytes.

9 Volume recognition structures

9.1 Volume Structure Descriptor

The Volume Structure Descriptor shall be recorded in the format shown in figure 22.

BP	Length	Name	Contents
0	1	Structure Type	`Uint8` (1/7.1.1)
1	5	Standard Identifier	bytes
6	1	Structure Version	`Uint8` (1/7.1.1)
7	2 041	Structure Data	bytes

Figure 2 - Generic Volume Structure Descriptor format

9.1.1 Structure Type (BP 0)

The number in this field shall specify the type of the Volume Structure Descriptor. The interpretation of the number shall be specified by the Standard or clause identified in the Standard Identifier field.

9.1.2 Standard Identifier (BP 1)

This field shall specify the interpretation of the Volume Structure Descriptor as shown in figure 23.

Identifier	Interpretation
“BEA01”	According to 2/9.2.
“BOOT2”	According to 2/9.4.
“CD001”	According to ECMA-119.
“CDW02”	According to ECMA-168.
“NSR02”	According to 3/9.1 of this ECMA Standard.
“TEA01”	According to 2/9.3.

Figure 3 - Volume Structure Descriptor interpretation

All other values are reserved for future standardisation.

9.1.3 Structure Version (BP 6)

The number in this field shall specify the version of the Volume Structure Descriptor. The interpretation of the number shall be specified by the Standard or clause identified in the Standard Identifier field.

9.1.4 Structure Data (BP 7)

The interpretation of this field shall be specified by the Standard or clause identified in the Standard Identifier field.

9.2 Beginning Extended Area Descriptor

The Beginning Extended Area Descriptorshall be recorded in the format shown in figure 24.

BP	Length	Name	Contents
0	1	Structure Type	Uint8 (1/7.1.1) = 0
1	5	Standard Identifier	bytes = “BEA01”
6	1	Structure Version	Uint8 (1/7.1.1) = 1
7	2 041	Structure Data	#00 bytes

Figure 4 - Beginning Extended Area Descriptor format

9.2.1 Structure Type (BP 0)

This field shall specify 0.

9.2.2 Standard Identifier (BP 1)

This field shall specify “BEA01”.

9.2.3 Structure Version (BP 6)

This field shall specify the version of this descriptor. The value 1 shall indicate the structure of Part 2.

9.2.4 Structure Data (BP 7)

This field shall be reserved for future standardisation and all bytes shall be set to #00.

9.3 Terminating Extended Area Descriptor

The Terminating Extended Area Descriptor shall be recorded in the format shown in figure 25.

BP	Length	Name	Contents
0	1	Structure Type	Uint8 (1/7.1.1) = 0
1	5	Standard Identifier	bytes = “TEA01”
6	1	Structure Version	Uint8 (1/7.1.1) = 1
7	2 041	Structure Data	#00 bytes

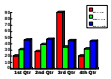

Figure 5 - Terminating Extended Area Descriptor format

9.3.1 Structure Type (BP 0)

This field shall specify 0.

9.3.2 Standard Identifier (BP 1)

This field shall specify “TEA01”.

9.3.3 Structure Version (BP 6)

This field shall specify the version of this descriptor. The value 1 shall indicate the structure of Part 2.

9.3.4 Structure Data (BP 7)

This field shall be reserved for future standardisation and all bytes shall be set to #00.

9.4 Boot Descriptor

The Boot Descriptor shall be recorded in the format shown in figure 26.

BP	Length	Name	Contents
0	1	Structure Type	`Uint8` (1/7.1.1) = 0
1	5	Standard Identifier	bytes = “BOOT2”
6	1	Structure Version	`Uint8` (1/7.1.1) = 1
7	1	Reserved	#00 byte
8	32	Architecture Type	`regid` (1/7.4)
40	32	Boot Identifier	`regid` (1/7.4)
72	4	Boot Extent Location	`Uint32` (1/7.1.5)
76	4	Boot Extent Length	`Uint32` (1/7.1.5)
80	8	Load Address	`Uint64` (1/7.1.7)
88	8	Start Address	`Uint64` (1/7.1.7)
96	12	Descriptor Creation Date and Time	`timestamp` (1/7.3)
108	2	Flags	`Uint16` (1/7.1.3)
110	32	Reserved	#00 bytes
142	1 906	Boot Use	bytes

Figure 6 - Boot Descriptor format

9.4.1 Structure Type (BP 0)

This field shall specify 0.

9.4.2 Standard Identifier (BP 1)

This field shall specify “BOOT2”.

9.4.3 Structure Version (BP 6)

This field shall specify the version of this descriptor. The value 1 shall indicate the structure of Part 2.

9.4.4 Reserved (BP 7)

This field shall be reserved for future standardisation and shall be set to #00.

9.4.5 Architecture Type (BP 8)

This field shall specify an identification of a system which can recognise and act upon the contents of the Boot Identifier field. If this field contains all #00 bytes, no such system is identified.

9.4.6 Boot Identifier (BP 40)

This field shall specify an identification of an implementation which can recognise and act upon the contents of the Boot Extent Location, Boot Extent Length, Load Address, Start Address and Boot Use fields. If this field contains all #00 bytes, no such implementation is identified.

9.4.7 Boot Extent Location (BP 72)

This field shall specify the address of an extent of the volume containing boot information. If the Boot Extent Length field contains 0, then no boot extent is specified and this field shall contain 0.

Note 2

If no boot extent is specified, the information needed to boot might be recorded in the Boot Use field.

9.4.8 Boot Extent Length (BP 76)

This field shall specify the length, in bytes, of the extent identified by the Boot Extent Location field.

9.4.9 Load Address (BP 80)

This field shall specify the memory address at which the information in the extent specified by the Boot Extent field should be copied.

9.4.10 Start Address (BP 88)

This field shall specify the memory address to which control should be transferred after the information specified by the Boot Extent field has been copied into memory.

9.4.11 Descriptor Creation Date and Time (BP 96)

This field shall specify the date and time of the day at which the information in this descriptor was recorded.

9.4.12 Flags (BP 108)

This field shall specify certain characteristics of the Boot Descriptor as shown in figure 2/7.

Bit	Interpretation
0	Erase: For any Boot Descriptor with the same contents of the Architecture Type and Boot Identifier fields as this Boot Descriptor and recorded in any lower numbered sectors of the volume recognition sequence than the sectors that this Boot Descriptor is recorded in: if set to ZERO, shall mean that this Boot Descriptor overrides any such Boot Descriptor; if set to ONE, shall mean that any such Boot Descriptor (including this Boot Descriptor) shall be ignored.
1-15	Shall be reserved for future standardisation and all bits shall be set to ZERO.

Figure 7 - Boot Descriptor characteristics

9.4.13 Reserved (BP 110)

This field shall be reserved for future standardisation and all bytes shall be set to #00.

9.4.14 Boot Use (BP 142)

This field shall be reserved for implementation use, and its contents are not specified by Part 2.

Note 3

The Boot Descriptor is designed to allow a generic boot program. Such a boot program would scan for Boot Descriptors with a matching Architecture Type (which might represent combinations of processor type and memory management), and after examining the Boot Identifier, which might encode the operating system type and options, present the user with a choice of operating systems to boot. As Part 2 cannot mandate any specific implementation behaviour, the recommended interpretation of the Boot Descriptor, that is, read an extent of sectors from the volume into memory at a specified location and then transfer execution to another specified location, is optional.

10 Levels of medium interchange

Part 2 specifies two levels of medium interchange. The level of a volume shall be that level specifying the most restrictions required to record the volume according to the specifications of Part 2.

10.1 Level 1

At level 1, the following restriction shall apply:

– The Boot Identifier field of a Boot Descriptor shall be different from the Boot Identifier field of all other Boot Descriptors having identical contents of the Architecture Type field.

10.2 Level 2

At level 2, no restrictions shall apply.

Section 3 - Requirements for systems for volume and boot block recognition

11 Requirements for the description of systems

Part 2 specifies that certain information shall be communicated between a user and an implementation. Each implementation that conforms to Part 2 shall have a description that identifies the means by which the user may supply or obtain such information.

Note 4

The specifics of the description and the means referred to above will vary from implementation to implementation. For example, an implementation might support two interfaces: a preferred, convenient interface which might vet user input, and a deprecated low level interface which allows any input specified by Part 2.

12 Requirements for an originating system

12.1 General

The implementation shall be capable of recording Beginning Extended Area Descriptors and Terminating Extended Area Descriptors as specified in Part 2 on a volume.

12.2 Optional access by user

12.2.1 Descriptors

If the implementation is capable of recording a Volume Structure Descriptor with the value “CD001” or “CDW02” or “NSR02” in the Standard Identifier field, the implementation shall record the descriptor according to the Standard or 2/9.1.2.

13 Requirements for a receiving system

13.1 General

The implementation shall be capable of interpreting Beginning Extended Area Descriptors and Terminating Extended Area Descriptors as specified in Part 2 on a volume.

13.2 Optional access by user

13.2.1 Descriptors

If the implementation is capable of interpreting a Volume Structure Descriptor with the value “CD001” or “CDW02” or “NSR02” in the Standard Identifier field, the implementation shall interpret the descriptor according to the Standard or 2/9.1.2.

Standard ECMA - 167

Volume and File Structure of Write-Once and Rewritable Media using Non-Sequential Recording for Information Interchange

Part 3 : Volume structure

Section 1 - General

1 Scope

Part 3 specifies a format and associated system requirements for volume structure by specifying:

- the attributes of a volume and the descriptors recorded on it;
- the relationship among volumes of a volume set;
- the attributes of a partition of a volume;
- the attributes of a logical volume and the descriptors recorded on it;
- levels of medium interchange;
- requirements for the processes which are provided within information processing systems, to enable information to be interchanged between different systems; for this purpose, it specifies the functions to be provided within systems which are intended to originate or receive media which conform to Part 3.

2 Parts references

See 1/2.

3 Part interface

This clause specifies the interface of Part 3 to other standards or Parts.

3.1 Input

Part 3 requires the specification of the following by another standard or Part.

- A standard for recording (see 1/5.10).
- The size of a logical sector (see 3/8.1.2) of a volume.
- If the volume is recorded according to Part 2, a volume recognition sequence specified by Part 2 shall contain the descriptor described in 3/9.1 recorded at least once.
- If the volume is recorded according to Part 2, the volume recognition space (see 2/8.2) shall be the entire volume.
- If the volume is recorded according to Part 2, the initial sector in the volume (see 2/3.1) shall be the first sector of the volume.
- Information to be recorded in the Partition Contents Use field of a Partition Descriptor (see 3/10.5.6).
- Information to be recorded in the Logical Volume Contents Use field of a Logical Volume Descriptor (see 3/10.6.7).

3.2 Output

Part 3 specifies the following which may be used by other standards or Parts.

- Volume sets of one or more volumes (see 3/8.6).
- A volume space for a volume (see 3/8.2).
- Logical sectors of a fixed size for a volume (see 3/8.1.2).
- Partitions (see 3/8.7).
- Logical volumes composed of partitions (see 3/8.8).
- Numeric identification of the partitions within a logical volume (see 3/8.8).
- Logical blocks of a fixed size for a logical volume
- The logical blocksize for a logical volume
- Attributes of a volume.

- Attributes of a logical volume
- Attributes of a partition.
- An indication that a volume may have been recorded to this Part (see 3/9.1).

4 Conformance

See 1/3.

5 Definitions

In addition to the definitions of Part 1 (see 1/5), the following definitions apply for Part 3.

5.1 anchor point

One of a specified set of logical sector numbers at which descriptors, that identify an extent of a Volume Descriptor Sequence, may be recorded.

5.2 Cyclic Redundancy Check (CRC)

A method for computing a signature of a sequence of bytes.

5.3 extent

A set of logical sectors whose logical sector numbers (see 3/8.1.3) form a continuous ascending sequence. The address, or location, of an extent is the first logical sector number in that sequence.

5.4 logical block

The unit of allocation of a logical volume.

5.5 logical sector

The unit of allocation of a volume.

5.6 logical volume

A nonempty set of partitions.

5.7 partition

An extent of logical sectors within a volume.

6 Notation

The notation of Part 1 (see 1/6) applies to Part 3.

7 Basic types

In addition to the basic types of Part 1 (see 1/7), the following basic types apply for Part 3.

7.1 Extent Descriptor

An Extent Descriptor, hereafter designated as `extent_ad`, shall be recorded in the format shown in figure 3/1.

RBP	Length	Name	Contents
0	4	Extent Length	`Uint32` (1/7.1.5)
4	4	Extent Location	`Uint32` (1/7.1.5)

Figure 1 - `extent_ad` format

7.1.1 Extent Length (RBP 0)

This field shall indicate the length of the extent, in bytes, identified by the Extent Location field. The length shall be less than 2^{30}. Unless otherwise specified, the length shall be an integral multiple of the logical sector size.

7.1.2 Extent Location (RBP 4)

This field shall specify the location of the extent, as a logical sector number. If the extent's length is 0, no extent is specified and this field shall contain 0.

7.2 Descriptor tag

Certain descriptors specified in Part 3 have a 16 byte structure, or `tag`, recorded at the start of the descriptor. The `tag` shall be recorded with the format shown in figure 32.

Note 1

There are two main motivations for using a generic tag structure. The first is that most descriptors need to handle common issues of CRCs and format versions. The second motivation is to support recovery after the medium has been damaged or corrupted in some (unspecified) way. With the tag described here, structures are self identifying and can be verified with very little context.

RBP	Length	Name	Contents
0	2	Tag Identifier	`Uint16` (1/7.1.3)
2	2	Descriptor Version	`Uint16` (1/7.1.3) = 2
4	1	Tag Checksum	`Uint8` (1/7.1.1)
5	1	Reserved	#00 byte
6	2	Tag Serial Number	`Uint16` (1/7.1.3)
8	2	Descriptor CRC	`Uint16` (1/7.1.3)
10	2	Descriptor CRC Length	`Uint16` (1/7.1.3)
12	4	Tag Location	`Uint32` (1/7.1.5)

Figure 2 - `tag` format

7.2.1 Tag Identifier (RBP 0)

This field shall specify an identification of the descriptor type. Type 0 shall specify that the format of this descriptor is not specified by Part 3. Types 1-7 and 9 are specified as shown in figure 3/ 3. Type 8 is specified identically in Part 3 and Part 4. Types 256-265 are specified in Part 4. All other types are reserved for future standardisation. The descriptor types specified by Part 3 are shown in figure 33.

Type	Interpretation
1	Primary Volume Descriptor (3/10.1)
2	Anchor Volume Descriptor Pointer (3/10.2)
3	Volume Descriptor Pointer (3/10.3)
4	Implementation Use Volume Descriptor (3/10.4)
5	Partition Descriptor (3/10.5)
6	Logical Volume Descriptor (3/10.6)
7	Unallocated Space Descriptor (3/10.8)
8	Terminating Descriptor (3/10.9 and 4/14.2)
9	Logical Volume Integrity Descriptor (3/10.10)

Figure 3 - Descriptor interpretation

7.2.2 Descriptor Version (RBP 2)

This field shall specify the version of this descriptor. The value 2 shall indicate the structure of Part 3.

7.2.3 Tag Checksum (RBP 4)

This field shall specify the sum modulo 256 of bytes 0-3 and 5-15 of the tag.

7.2.4 Reserved (RBP 5)

This field shall be reserved for future standardisation and shall be set to 0.

7.2.5 Tag Serial Number (RBP 6)

This field shall specify an identification of a set of descriptors. If the field contains 0, then no such identification is specified.

Note 2

This field can be used to distinguish between groups of descriptors. For example, when reusing rewritable media, an implementation might choose a different serial number from the previous use when initialising a volume. Thus, a disaster recovery mechanism can avoid recovering prior and unintended data. The only alternative to this scheme would be to force volume initialisation to clear the volume.

7.2.6 Descriptor CRC (RBP 8)

This field shall specify the CRC of the bytes of the descriptor starting at the first byte after the descriptor tag . The number of bytes shall be specified by the Descriptor CRC Length field. The CRC shall be 16 bits long and be generated by the CRC-ITU-T polynomial (see ITU-T V.41):

$$x^{16} + x^{12} + x^5 + 1$$

Note 3

As an example, the CRC of the three bytes #70 #6A #77 is #3299. Implementations can avoid calculating the CRC by setting the Descriptor CRC Length to 0, as then the Descriptor CRC shall be 0.

7.2.7 Descriptor CRC Length (RBP 10)

This field specifies how many bytes were used in calculating the Descriptor CRC.

7.2.8 Tag Location (RBP 12)

This field shall specify the number of the logical sector containing the first byte of the descriptor.

Note 1

The location of the tag may appear to be redundant but its primary purpose is to make it extremely likely that if the first 16 bytes of a logical sector or logical block is a consistent descriptor tag, then it is a descriptor tag.

Section 2 - Requirements for the medium for volume structure

8 Volume structure

8.1 Arrangement of information on a volume

8.1.1 Sector numbers

Each sector of a volume shall be identified by a unique sector number. Sector numbers shall be consecutive integers assigned in an ascending sequence, in the order of ascending physical address of the volume as specified in the relevant standard for recording (see 1/5.10). Sector number 0 shall be assigned to the sector having the lowest physical address of the volume.

8.1.2 Logical sector

The sectors of a volume shall be organised into logical sectors of equal length. The length of a logical sector shall be referred to as the logical sector size and shall be an integral multiple of 512 bytes. The logical sector size shall be not less than the size of the smallest sector of the volume. Each logical sector shall begin in a different sector, starting with the sector having the next higher sector number than that of the last sector constituting the previous, if any, logical sector of the volume. The first byte of a logical sector shall be the first byte of the sector in which it begins, and if the size of this sector is smaller than the logical sector size, then the logical sector shall comprise a sequence of constituent sectors with consecutive ascending sector numbers.

8.1.3 Logical sector numbers

Each logical sector of a volume shall be identified by a unique logical sector number. Logical sector numbers shall be consecutive integers assigned in ascending sequence, in the order of ascending sector numbers of the volume. Logical sector number 0 shall be assigned to the logical sector beginning in sector number 0. The largest logical sector number of a volume shall be greater than 256.

8.1.3.1 Recording of logical sectors

Any unrecorded constituent sector of a logical sector shall be interpreted as containing all #00 bytes. Within the sector containing the last byte of a logical sector, the interpretation of any bytes after that last byte is not specified by this Part.

A logical sector is unrecorded if the standard for recording allows detection that a sector has been unrecorded and all of the logical sector's constituent sectors are unrecorded. A logical sector should either be completely recorded or unrecorded.

8.2 Volume space

The information on a volume shall be recorded in the set of all logical sectors in a volume. This set shall be referred to as the volume space of the volume. The bytes in the volume space shall be numbered with consecutive integers assigned in ascending sequence starting with 0. Let s be the number of bytes in a logical sector; then byte b of the volume space is byte $rem(b,s)$ of logical sector $ip(b/s)$.

8.3 Volume descriptors

Characteristics of the volume shall be specified by volume descriptors recorded in Volume Descriptor Sequences as described in 3/8.4.2.

A volume descriptor shall be one of the following types:

- Primary Volume Descriptor (see 3/10.1)
- Implementation Use Volume Descriptor (see 3/10.4)
- Partition Descriptor (see 3/10.5)
- Logical Volume Descriptor (see 3/10.6)
- Unallocated Space Descriptor (see 3/10.8)

8.4 Volume Descriptor Sequence

8.4.1 Contents of a Volume Descriptor Sequence

A Volume Descriptor Sequence shall contain one or more Primary Volume Descriptors. A Primary Volume Descriptor shall identify the volume and the volume set to which it belongs, the sequence number of the volume within the volume set, attributes of the volume, and the character sets used in recording the contents of certain fields within the Primary Volume Descriptor. Each Primary Volume Descriptor shall have an assigned Primary Volume Descriptor Number. Only one prevailing Primary Volume Descriptor (see 3/8.4.3) of a Volume Descriptor Sequence shall have a Primary Volume Descriptor Number of 0.

A Volume Descriptor Sequence shall contain zero or more Implementation Use Volume Descriptors. An Implementation Use Volume Descriptor shall identify an implementation and contain information for that implementation's use.

A Volume Descriptor Sequence shall contain zero or more Partition Descriptors. A Partition Descriptor shall specify a partition, attributes of the partition and an identification of the partition, referred to as the partition number.

A Volume Descriptor Sequence shall contain zero or more Logical Volume Descriptors. A Logical Volume Descriptor shall specify an identification of the logical volume, the logical block size of the logical volume, identification of the partitions comprising the logical volume and attributes of the logical volume.

A Volume Descriptor Sequence shall contain zero or more Unallocated Space Descriptors. An Unallocated Space Descriptor shall identify volume space available for allocating to partitions or for recording the Volume Descriptor Sequences of the volume.

Each volume descriptor shall have an assigned Volume Descriptor Sequence Number. All volume descriptors with identical Volume Descriptor Sequence Numbers shall have identical contents.

Note 5

Typically, an originating system will chose a new Volume Descriptor Sequence Number by adding 1 to the largest such number seen when scanning the Volume Descriptor Sequence.

8.4.2 Recording of the Volume Descriptor Sequence

A Volume Descriptor Sequence shall be recorded as a sequence of extents of logical sectors in the volume space. Any trailing sectors (see figure 3/4) shall be available for the recording of descriptors.

An extent of a Volume Descriptor Sequence shall be recorded according to the schema shown in figure 3/4.

```
[Volume Descriptor Sequence extent]{
            <volume descriptor>0+
        [Terminator]{
            <Volume Descriptor Pointer>
          | <Terminating Descriptor>
          | <unrecorded logical sector>
        } <trailing logical sector>0+
}
```

Figure 4 - Volume Descriptor Sequence schema

An extent of a Volume Descriptor Sequence shall be identified by an Anchor Volume Descriptor Pointer (see 3/10.2) recorded at two or more anchor points (see 3/8.4.2.1). Each, if any, subsequent extent in the Volume Descriptor Sequence shall be identified by a Volume Descriptor Pointer recorded in the previous extent of the sequence.

An Anchor Volume Descriptor Pointer shall identify the Main Volume Descriptor Sequence and may identify a Reserve Volume Descriptor Sequence (see 3/8.4.2.2). All Volume Descriptor Sequences specified by Anchor Volume Descriptor Pointers shall be equivalent (see 3/8.4.2.3).

8.4.2.1 Anchor points

Let *k* be *ip(n/*59*)*, where *n* is the largest logical sector number in the volume space. Anchor points shall be at two or more of the following logical sector numbers: 256, *n*–256, *n* and all the nonzero integral multiples of *k* not greater than *n*.

Note 6

The value 59 was chosen as a number near 64 that was unlikely to be periodic with respect to the geometry of the underlying medium.

8.4.2.2 Reserve Volume Descriptor Sequence Set

A Reserve Volume Descriptor Sequence may be recorded on a volume. A Reserve Volume Descriptor Sequence, if any, shall be identified by an Anchor Volume Descriptor Pointer. If any Anchor Volume Descriptor Pointer of the volume identifies a Reserve Volume Descriptor Sequence, then all Anchor Volume Descriptor Pointers of the volume shall identify a Reserve Volume Descriptor Sequence. If the Reserve Volume Descriptor Sequence is identified, it shall specify a Volume Descriptor Sequence equivalent to the Main Volume Descriptor Sequence (see 3/8.4.2.3). There shall be no logical sector which belongs to both an extent of the Main Volume Descriptor Sequence and an extent of the Reserve Volume Descriptor Sequence.

8.4.2.3 Equivalent Volume Descriptor Sequences

The equivalence of two Volume Descriptor Sequences shall be determined by calculating a canonical form for each of the Volume Descriptor Sequences, and if both the canonical forms are identical, then the two Volume Descriptor Sequences specify equivalent sets of volume descriptors. The canonical form of a Volume Descriptor Sequence shall be constructed by performing the following steps in sequence:

- discard any Volume Descriptor Pointers
- discard all but one descriptor for each Volume Descriptor Sequence Number
- set the Tag Checksum, Descriptor CRC, Descriptor CRC Length, and Tag Location fields of the Descriptor Tag field in each descriptor to 0
- sort the remaining descriptors as byte sequences
- catenate the descriptors in sorted order

8.4.3 Prevailing descriptors

Within each of the following classes of descriptors:

- Primary Volume Descriptors, each of which has the same contents of the corresponding Volume Identifier, Volume Set Identifier and Descriptor Character Set fields
- Partition Descriptors with identical Partition Numbers
- Logical Volume Descriptors, each of which has the same contents of the corresponding Logical Volume Identifier and Descriptor Character Set fields
- Unallocated Space Descriptors

the one with the highest Volume Descriptor Sequence Number shall be used. This instance shall be referred to as the prevailing instance.

8.4.4 Recording of descriptors

All the descriptors in Part 3 whose format is specified with Byte Positions (BP) shall be recorded so that the first byte of the descriptor coincides with the first byte of a logical sector

The descriptors in Part 3 whose format is specified with Relative Byte Positions (RBP) have no restrictions on where they may be recorded within a logical sector, except that their location within a descriptor shall be specified in the description of the applicable descriptor.

When the descriptors described in Part 3 are recorded in a logical sector , all space, if any, after the end of the last descriptor up to the end of the logical sector is reserved for future standardisation and shall be recorded as all #00 bytes.

Note 7

Most of the descriptors specified in Part 3 have a length of 512 bytes.

8.5 Allocation of the volume space

The logical sector is the unit of allocation for the volume space . Volume space may be allocated for the recording of Volume Descriptor Sequence s, or Anchor Volume Descriptor Pointer s, or may be allocated to partitions. This allocation shall be done from the unallocated volume space which shall be specified as extents of logical sectors by the prevailing instance of the Unallocated SpaceDescriptor (see 3/10.8).

Volume Descriptor Sequence s and Anchor Volume Descriptor Pointer s shall not be recorded in volume space that has been allocated to a partition.

Note 8

Implementations should not assume that the sum of the allocated logical sectors and the unallocated logical sectors in a volume equals the size of the volume space. Parts of the volume space might be unallocated for several reasons, including media defects or for use by processes external to Part 3.

8.6 Volume set

A volume set shall consist of one or more volumes having a volume set identification common to all volumes in the volume set. The volumes in a volume set shall be numbered with consecutive integers assigned in an ascending sequence starting from 1. This number shall be the assigned volume sequence number of the volume.

Each prevailing Primary Volume Descriptor recorded on a volume contains a volume set identification consisting of the contents of the Volume Set Identifier and Descriptor Character Set fields, a volume identification consisting of the contents of the Volume Identifier and Descriptor Character Set fields, and specifies whether that volume set identification is common to each volume of the volume set. Exactly one of the volume set identifications specified on a volume shall be marked as being common to each volume of the volume set (see 3/10.1.21). The same volume identification shall not be specified by more than one volume of a volume set.

8.7 Partition

A partition is an extent of a volume and shall be identified by a Partition Number in the range 0 to 65 535 inclusive. The information about a partition shall be recorded in a Partition Descriptor. The prevailing instance of the Partition Descriptor with a specific Partition Number shall specify whether volume space has been allocated to the partition and may specify an identification of the partition's contents.

Note 9

Partitions may overlap. This allows media to be initialised with several predefined partition definitions of varying sizes and locations. A user can then simply select a set of nonoverlapping partitions to use. In general, it is inadvisable to use file systems on overlapping partitions.

8.8 Logical volume

A Logical Volume Descriptor specifies a logical volume identification, the logical block size of the logical volume, and an ordered list of partitions comprising the logical volume. The partitions of a logical volume may be on different volumes of a volume set

The partitions in a logical volume shall be numbered with consecutive integers assigned in an ascending sequence starting from 0. This number shall be the assigned partition reference number and shall be the numeric identification referred to in 3/3.2.

A logical volume shall be described by the prevailing instance of a Logical Volume Descriptor recorded on the volume with the highest volume sequence number in a volume set . Each Logical Volume Descriptor recorded on a volume set contains a logical volume identification consisting of the contents of the Logical Volume Identifier and Descriptor Character Set fields. More than one logical volume may be recorded on a volume set. Logical Volume Descriptors for all the logical volumes in a volume set shall be recorded in the volume with the highest volume sequence number.

Note 10

A logical volume provides a segmented address space that can span multiple partitions and volumes of a volume set. As a consequence of this, a logical volume may only belong to one volume set.

8.8.1 Logical blocks

The logical sectors of a partition in a logical volume shall be organised into logical blocks of equal length. The length of a logical block shall be referred to as the logical block size and shall be an integral multiple of 512 bytes not less than the size of the logical sector of the volume. The logical block size of each partition of a logical volume shall be the same.

Each logical block shall begin in a different logical sector, and shall start with the logical sector having the next higher logical sector number than that of the last logical sector constituting the previous, if any, logical block of the partition. The first byte of a logical block shall be the first byte of the logical sector in which it begins, and if the logical sector size is smaller than the logical block size, then the logical block shall comprise a sequence of constituent logical sectors with consecutive ascending logical sector numbers. Within the logical sector containing the last byte of a logical block, the interpretation of any bytes after that last byte is not specified by this Part.

Each logical block of a partition shall be identified by a unique logical block number. Logical block numbers shall be consecutive integers assigned in ascending sequence. The logical block containing logical sector 0 shall have a logical block number of 0.

8.8.2 Logical volume integrity

Logical volume integrity describes the status of the information recorded on a logical volume. This status shall be specified by a Logical Volume Integrity Descriptor (see 3/10.10).

The Logical Volume Integrity Descriptors for a logical volume shall be recorded in a Logical Volume Integrity Sequence which shall be recorded as a sequence of extents. The first extent shall be specified by the prevailing Logical Volume Descriptor for the logical volume. Succeeding extents, if any, shall be specified by a Logical Volume Integrity Descriptor. Processing of an extent of Logical Volume Integrity Descriptors shall be as if the descriptors were processed in order of ascending order of their addresses and processing was terminated by an unrecorded logical sector, or a Terminating Descriptor (see 3/10.9) or after a descriptor specifying a subsequent extent. After processing all such extents, the last Logical Volume Integrity Descriptor processed shall be used and shall be referred to as the prevailing Logical Volume Integrity Descriptor.

The status of a logical volume shall be specified by the prevailing Logical Volume Integrity Descriptor as follows:

- An Open Integrity Descriptor shall be recorded before any data is recorded in the logical volume since the last Close Integrity Descriptor, if any, was recorded
- A Close Integrity Descriptor may be recorded only after the data recorded on the logical volume is in some consistent form not specified by this Part

9 Volume recognition structures

9.1 NSR Descriptor

The NSR Descriptor shall be recorded in the format shown in figure 35.

Note 11

This descriptor only indicates that a volume may have been recorded to Part 3; in particular, see 3/3.1 and 3/3.2.

BP	Length	Name	Contents
0	1	Structure Type	`Uint8` (1/7.1.1) = 0
1	5	Standard Identifier	bytes = “NSR02”
6	1	Structure Version	`Uint8` (1/7.1.1) = 1
7	1	Reserved	#00 byte
8	2 040	Structure Data	#00 bytes

Figure 5 - NSR Descriptor format

9.1.1 Structure Type (BP 0)

This field shall specify 0.

9.1.2 Standard Identifier (BP 1)

This field shall specify “NSR02”.

9.1.3 Structure Version (BP 6)

This field shall specify the version of this descriptor. The value 1 shall indicate the structure of Part 3.

9.1.4 Reserved (BP 7)

This field shall be reserved for future standardisation and shall be set to 0.

9.1.5 Structure Data (BP 8)

This field shall be reserved for future standardisation and all bytes shall be set to #00.

10 Volume data structures

10.1 Primary Volume Descriptor

The Primary Volume Descriptor shall identify a volume and certain attributes of that volume. It shall be recorded in the format shown in figure 36.

BP	Length	Name	Contents
0	16	Descriptor Tag	`tag` (3/7.2) (Tag=1)
16	4	Volume Descriptor SequenceNumber	`Uint32` (1/7.1.5)
20	4	Primary Volume DescriptorNumber	`Uint32` (1/7.1.5)
24	32	Volume Identifier	`dstring` 1/7.2.12)
56	2	Volume Sequence Number	`Uint16` (1/7.1.3)
58	2	Maximum Volume Sequence Number	`Uint16` (1/7.1.3)
60	2	Interchange Level	`Uint16` (1/7.1.3)
62	2	Maximum Interchange Level	`Uint16` (1/7.1.3)
64	4	Character Set List	`Uint32` (1/7.1.5)
68	4	Maximum Character Set List	`Uint32` (1/7.1.5)
72	128	Volume Set Identifier	`dstring` (1/7.2.12)
200	64	Descriptor Character Set	`charspec` (1/7.2.1)
264	64	Explanatory Character Set	`charspec` (1/7.2.1)
328	8	Volume Abstract	`extent_ad` (3/7.1)
336	8	Volume Copyright Notice	`extent_ad` (3/7.1)
344	32	Application Identifier	`regid` (1/7.4)
376	12	Recording Date and Time	`timestamp` (1/7.3)
388	32	Implementation Identifier	`regid` (1/7.4)
420	64	Implementation Use	bytes
484	4	Predecessor Volume Descriptor Sequence Location	`Uint32` (1/7.1.5)
488	2	Flags	`Uint16` (1/7.1.3)
490	22	Reserved	#00 bytes

Figure 6 - Primary Volume Descriptor format

10.1.1 Descriptor Tag (BP 0)

The Tag Identifier field of the `tag` (3/7.2) for this descriptor shall contain 1.

10.1.2 Volume Descriptor Sequence Number (BP 16)

This field shall specify the Volume Descriptor Sequence Number for this descriptor.

10.1.3 Primary Volume Descriptor Number (BP 20)

This field shall specify the assigned Primary Volume Descriptor Number for this Primary Volume Descriptor.

10.1.4 Volume Identifier (BP 24)

This field shall specify an identification of the volume.

10.1.5 Volume Sequence Number (BP 56)

This field shall specify the ordinal number of the volume in the volume set of which the volume is a member.

10.1.6 Maximum Volume Sequence Number (BP 58)

This field shall specify the ordinal number of the volume in the volume set with the largest assigned volume sequence number at the time this descriptor was recorded. If this field contains 0, there is no such identification.

10.1.7 Interchange Level (BP 60)

This field shall specify the current level of medium interchange (3/11) of the volume described by this descriptor.

10.1.8 Maximum Interchange Level (BP 62)

This field shall specify the maximum value that may be specified for the Interchange Level field of this descriptor.

10.1.9 Character Set List (BP 64)

This field shall identify the character sets specified by any field, whose contents are specified to be a `charspec` (1/7.2.1), of any descriptor specified in Part 3 and recorded on the volume described by this descriptor.

10.1.10 Maximum Character Set List (BP 68)

The Character Set List field in this descriptor shall not specify a character set (see 1/7.2.11) not specified by the Maximum Character Set List field.

Note 12

The Interchange Level, Maximum Interchange Level, Character Set List and Maximum Character Set List fields permit an implementation to:

- *determine whether it can process all of the information on the volume.*
- *restrict the recording of information on the volume so that the volume does not exceed the level given in the Maximum Interchange Level field.*
- *restrict the recording of information on the volume so that all character sets recorded belong to the Maximum Character Set List field.*

This allows a user to create a volume that can be processed when it is returned to the user.

10.1.11 Volume Set Identifier (BP 72)

This field shall specify an identification of the volume set of which the volume is a member.

10.1.12 Descriptor Character Set (BP 200)

This field shall specify the d-characters (1/7.2) allowed in the Volume Identifier and Volume Set Identifier fields.

10.1.13 Explanatory Character Set (BP 264)

This field shall specify how to interpret the contents of the Volume Abstract and Volume Copyright Notice extents.

10.1.14 Volume Abstract (BP 328)

This field shall specify an extent of logical sectors containing an abstract for this volume. If the extent's length is 0, no abstract is specified.

10.1.15 Volume Copyright Notice (BP 336)

This field shall specify an extent of logical sectors containing a copyright notice for this volume. If the extent's length is 0, no copyright notice is specified.

10.1.16 Application Identifier (BP 344)

This field shall specify an identification of an application. If this field contains all #00 bytes, then no such application is identified.

10.1.17 Recording Date and Time (BP 376)

This field shall indicate the date and time of the day at which this descriptor was recorded.

10.1.18 Implementation Identifier (BP 388)

This field shall specify an identification of an implementation which can recognise and act upon the contents of the Implementation Use field. If this field contains all #00 bytes, then no such implementation is identified. The scope of this `regid` includes the contents of all descriptors, other than Implementation Use Volume Descriptors, in the Volume Descriptor Sequence in which the Primary Volume Descriptor is recorded.

10.1.19 Implementation Use (BP 420)

This field shall be reserved for implementation use. Its content is not specified by this Standard.

10.1.20 Predecessor Volume Descriptor Sequence Location (BP 484)

This field shall specify the address of the extent (in logical sectors) in which the immediately preceding extent of the Volume Descriptor Sequence of the volume is recorded. If this field contains 0, it shall mean that no such extent is identified.

Note 13

This is intended for use in disaster recovery.

10.1.21 Flags (BP 488)

This field shall specify certain characteristics of this Primary Volume Descriptor as shown in figure 3/7.

Bit	Interpretation
0	Volume Set Identification: If set to ZERO, shall mean that the volume set identification in this descriptor need not be common among all volumes in the volume set that this volume belongs to; If set to ONE, shall mean that the volume set identification in this descriptor is common among all volumes in the volume set that this volume belongs to.
1-15	Shall be reserved for future standardisation and all bits shall be set to ZERO.

Figure 7 - Primary Volume Descriptor Characteristics

10.1.22 Reserved (BP 490)

This field shall be reserved for future standardisation and all bytes shall be set to #00.

10.2 Anchor Volume Descriptor Pointer

The Anchor Volume Descriptor Pointer shall specify an extent of the Main and Reserve Volume Descriptor Sequences recorded on the volume. It shall be recorded in the format shown in figure 3/8.

BP	Length	Name	Contents
0	16	Descriptor Tag	`tag` (3/7.2) (Tag=2)
16	8	Main Volume Descriptor Sequence Extent	`extent_ad` (3/7.1)
24	8	Reserve Volume Descriptor Sequence Extent	`extent_ad` (3/7.1)
32	480	Reserved	#00 bytes

Figure 8 - Anchor Volume Descriptor Pointer format

10.2.1 Descriptor Tag (BP 0)

The Tag Identifier field of the `tag` (3/7.2) for this descriptor shall contain 2.

10.2.2 Main Volume Descriptor Sequence Extent (BP 16)

This field shall specify an extent of the Main Volume Descriptor Sequence.

Note 14

The extent specifies allocation rather than recording; that is, space that may be available for recording rather than what is actually recorded. The extent need not be completely recorded.

10.2.3 Reserve Volume Descriptor Sequence Extent (BP 24)

This field shall specify an extent of the Reserve Volume Descriptor Sequence. If the extent's length is 0, no such extent is specified.

Note 15

The extent specifies allocation rather than recording; that is, space that may be available for recording rather than what is actually recorded. The extent need not be completely recorded.

10.2.4 Reserved (BP 32)

This field shall be reserved for future standardisation and all bytes shall be set to #00.

10.3 Volume Descriptor Pointer

The Volume Descriptor Pointer shall specify an extent of a Volume Descriptor Sequence recorded on the volume. It shall be recorded in the format shown in figure 39.

BP	Length	Name	Contents
0	16	Descriptor Tag	`tag` (3/7.2) (Tag=3)
16	4	Volume Descriptor Sequence Number	`Uint32` (1/7.1.5)
20	8	Next Volume Descriptor Sequence Extent	`extent_ad` (3/7.1)
28	484	Reserved	#00 bytes

Figure 9 - Volume Descriptor Pointer format

10.3.1 Descriptor Tag (BP 0)

The Tag Identifier field of the `tag` (3/7.2) for this descriptor shall contain 3.

10.3.2 Volume Descriptor Sequence Number (BP 16)

This field shall specify the Volume Descriptor Sequence Number for this descriptor.

10.3.3 Next Volume Descriptor Sequence Extent (BP 20)

This field shall specify the next extent in the Volume Descriptor Sequence. If the extent's length is 0, no such extent is specified.

Note 16

The extent specifies allocation rather than recording; that is, space that may be available for recording rather than what is actually recorded. The extent need not be completely recorded.

10.3.4 Reserved (BP 28)

This field shall be reserved for future standardisation and all bytes shall be set to #00.

10.4 Implementation Use Volume Descriptor

The Implementation Use Volume Descriptor shall identify an implementation which can recognise and act upon the contents of this descriptor's Implementation Use field. It shall be recorded in the format shown in figure 40.

BP	Length	Name	Contents
0	16	Descriptor Tag	`tag` (3/7.2) (Tag=4)
16	4	Volume Descriptor Sequence Number	`Uint32` (1/7.1.5)
20	32	Implementation Identifier	`regid` (1/7.4)
52	460	Implementation Use	bytes

Figure 10 - Implementation Use Volume Descriptor format

10.4.1 Descriptor Tag (BP 0)

The Tag Identifier field of the `tag` (3/7.2) for this descriptor shall contain 4.

10.4.2 Volume Descriptor Sequence Number (BP 16)

This field shall specify the Volume Descriptor Sequence Number for this descriptor.

10.4.3 Implementation Identifier (BP 20)

This field shall specify an identification of an implementation which can recognise and act upon the contents of the Implementation Use field. If this field contains all #00 bytes, then no such implementation is identified.

10.4.4 Implementation Use (BP 52)

This field shall be reserved for implementation use. Its content is not specified by this ECMA Standard.

10.5 Partition Descriptor

The Partition Descriptor shall specify the size and location of a partition and shall be recorded in the format shown in figure 3/11.

BP	Length	Name	Contents
0	16	Descriptor Tag	`tag` (3/7.2) (Tag=5)
16	4	Volume Descriptor Sequence Number	`Uint32` (1/7.1.5)
20	2	Partition Flags	`Uint16` (1/7.1.3)
22	2	Partition Number	`Uint16` (1/7.1.3)
24	32	Partition Contents	`regid` (1/7.4)
56	128	Partition Contents Use	bytes
184	4	Access Type	`Uint32` (1/7.1.5)
188	4	Partition Starting Location	`Uint32` (1/7.1.5)
192	4	Partition Length	`Uint32` (1/7.1.5)
196	32	Implementation Identifier	`regid` (1/7.4)
228	128	Implementation Use	bytes
356	156	Reserved	#00 bytes

Figure 11 - Partition Descriptor format

10.5.1 Descriptor Tag (BP 0)

The Tag Identifier field of the `tag` (3/7.2) for this descriptor shall contain 5.

10.5.2 Volume Descriptor Sequence Number (BP 16)

This field shall specify the Volume Descriptor Sequence Number for this descriptor.

10.5.3 Partition Flags (BP 20)

This field shall specify certain characteristics of the partition as shown in figure 3/12.

Bit	Interpretation
0	Allocation: If set to ZERO, shall mean that volume space has not been allocated for this partition; If set to ONE, shall mean that volume space has been allocated for this partition.
1-15	Shall be reserved for future standardisation and all bits shall be set to ZERO.

Figure 12 - Partition characteristics

10.5.4 Partition Number (BP 22)

This field shall specify the numeric identifier for the partition.

Note 17

The Partition Number may be 0.

10.5.5 Partition Contents (BP 24)

This field shall specify an identification of how to interpret the contents of the partition. The identifications specified by Part 3 are given in figure 3/13. Other identifications shall be specified according to 1/7.4.

Contents	Interpretation
"+FDC01"	As if it were a volume recorded according to ECMA-107.
"+CD001"	As if it were a volume recorded according to ECMA-119
"+CDW02"	As if it were a volume recorded according to ECMA-168.
"+NSR02"	According to Part 4 of this ECMA Standard.

Figure 13 - Partition content interpretation

10.5.6 Partition Contents Use (BP 56)

This field shall specify information required for the interpretation of the information recorded on the partition identified by this Partition Descriptor. The contents of this field shall be specified by the relevant standard for the interpretation of the information recorded on the partition.

10.5.7 Access Type (BP 184)

This field shall specify the access methods which are permitted on the logical sectors of the partition described by this Partition Descriptor. The access types are given in figure 14.

Type	Interpretation
0	The type of access is not specified by this field.
1	Read only: there shall be no restriction on reading logical sectors; logical sectors shall not be recorded.
2	Write once: there shall be no restriction on reading logical sectors; logical sectors shall only be recorded once.
3	Rewritable: there shall be no restriction on reading logical sectors; logical sectors may require preprocessing before recording.
4	Overwritable: there shall be no restriction on reading or recording logical sectors.
5 and above	Reserved for future standardisation.

Figure 14 - Access interpretation

10.5.8 Partition Starting Location (BP 188)

This field shall specify the logical sectornumber at which the partition begins.

10.5.9 Partition Length (BP 192)

This field shall specify the number of logical sectors which comprise the partition.

10.5.10 Implementation Identifier (BP 196)

This field shall specify an identification of an implementation which can recognise and act upon the contents of the Implementation Use field. If this field contains all #00 bytes, then no such implementation is identified.

Note 18

The scope of this `regid` *does not include the contents of the partition.*

10.5.11 Implementation Use (BP 228)

This field shall be reserved for implementation use and its contents are not specified by this ECMA Standard.

10.5.12 Reserved (BP 356)

This field shall be reserved for future standardisation and all bytes shall be set to #00.

10.6 Logical Volume Descriptor

The Logical Volume Descriptorshall be recorded in the format shown in figure 315.

BP	Length	Name	Contents
0	16	Descriptor Tag	`tag` (3/7.2) (Tag=6)
16	4	Volume Descriptor SequenceNumber	`Uint32` (1/7.1.5)
20	64	Descriptor Character Set	`charspec` (1/7.2.1)
84	128	Logical Volume Identifier	`dstring` (1/7.2.12)
212	4	Logical BlockSize	`Uint32` (1/7.1.5)
216	32	Domain Identifier	`regid` (1/7.4)
248	16	Logical Volume Contents Use	bytes
264	4	Map Table Length (=MT_L)	`Uint32` (1/7.1.5)
268	4	Number of Partition Maps (=N_PM)	`Uint32` (1/7.1.5)
272	32	Implementation Identifier	`regid` (1/7.4)
304	128	Implementation Use	bytes
432	8	Integrity Sequence Extent	`extent_ad` (3/7.1)
440	MT_L	Partition Maps	bytes

Figure 15 - Logical Volume Descriptorformat

10.6.1 Descriptor Tag (BP 0)

The Tag Identifier field of the `tag` (3/7.2) for this descriptor shall contain 6.

10.6.2 Volume Descriptor Sequence Number (BP 16)

This field shall specify the Volume Descriptor Sequence Number for this descriptor.

10.6.3 Descriptor Character Set (BP 20)

This field shall specify the d-characters (1/7.2) allowed in the Logical Volume Identifier field.

10.6.4 Logical Volume Identifier (BP 84)

This field shall specify an identification of the logical volume.

10.6.5 Logical Block Size (BP 212)

This field shall specify the size of a logical blockin bytes.

10.6.6 Domain Identifier (BP 216)

This field shall specify an identification of a domain which shall specify rules on the use of, and restrictions on, certain fields in descriptors subject to agreement between the originator and recipient of the medium. If this field contains all #00 bytes, then no such domain is identified. The scope of this `regid` (1/7.4) shall include all information recorded in the logical volume described by this descriptor, and shall include the scope of the Implementation Identifier field.

10.6.7 Logical Volume Contents Use (BP 248)

This field shall specify information required for the interpretation of the information recorded on the logical volume identified by this Logical Volume Descriptor. The contents of this field shall be specified by the relevant standard for the interpretation of the information recorded on the logical volume.

10.6.8 Map Table Length (=MT_L) (BP 264)

This field shall specify the length of the Partition Maps field in bytes.

10.6.9 Number of Partition Maps (=N_PM) (BP 268)

This field shall specify the number of Partition Maps recorded in the Partition Maps field.

10.6.10 Implementation Identifier (BP 272)

This field shall specify an identification of an implementation which can recognise and act upon the contents of the Implementation Use field. If this field contains all #00 bytes, then no such implementation is identified.

The scope of this `regid` includes the contents of any Partition Descriptors identified by Type 1 Partition Maps recorded in the Partition Maps field.

10.6.11 Implementation Use (BP 304)

This field shall be reserved for implementation use and its contents are not specified by this ECMA Standard.

10.6.12 Integrity Sequence Extent (BP 432)

This field shall specify the first extent of the Logical Volume Integrity Sequence. The extent shall be within the volume in which this descriptor is recorded. If N_PM is 0, then the extent's length may be 0. If the extent's length is 0, then no such extent is specified.

10.6.13 Partition Maps (BP 440)

This field shall contain N_PM Partition Maps recorded contiguously starting at the first byte of the field. The Partition Maps may be of different types. The length of the Partition Maps shall not exceed MT_L bytes and any unused bytes shall be set to #00.

As specified by 3/8.4.4, the remainder of the last logical sector comprising the Logical Volume Descriptor shall be recorded with #00 bytes.

10.7 Partition maps

10.7.1 Generic partition map

A partition map shall be recorded in the format shown in figure 316.

RBP	Length	Name	Contents
0	1	Partition Map Type	`Uint8` (1/7.1.1)
1	1	Partition Map Length (= PM_L)	`Uint8` (1/7.1.1)
2	PM_L-2	Partition Mapping	bytes

Figure 16 - Generic partition map format

10.7.1.1 Partition Map Type (RBP 0)

The number in this field shall specify the type of the partition map. The types are given in figure 317.

Type	Interpretation
0	Shall mean that the type of the partition map is not specified by this field.
1	Shall mean that the partition map is a Type 1 Partition Map (see 3/10.7.2).
2	Shall mean that the partition map is a Type 2 Partition Map (see 3/10.7.3).
3-255	Reserved for future standardisation.

Figure 17 - Partition maps

10.7.1.1.1 Partition Map Length (= PM_L) (RBP 1)

This field shall specify the length, in bytes, of this partition map, including the Partition Map Type and Partition Map Length fields.

10.7.1.1.2 Partition Mapping (RBP 2)

The interpretation of this field shall be specified by the standard or clause identified in the Partition Map Type field, or shall be subject to agreement between the originator and recipient of the medium if the number in the Partition Map Type field is 0.

10.7.2 Type 1 Partition Map

This map type identifies a partition on a volume in the volume set on which the logical volume is recorded. It shall be recorded in the format shown in figure 318.

RBP	Length	Name	Contents
0	1	Partition Map Type	`Uint8` (1/7.1.1) = 1
1	1	Partition Map Length	`Uint8` (1/7.1.1) = 6
2	2	Volume Sequence Number	`Uint16` (1/7.1.3)
4	2	Partition Number	`Uint16` (1/7.1.3)

Figure 18 - Type 1 Partition Map format

10.7.2.1 Partition Map Type (RBP 0)

This field shall specify 1.

10.7.2.2 Partition Map Length (RBP 1)

This field shall specify 6.

10.7.2.3 Volume Sequence Number (RBP 2)

This field specifies the volume, in the volume set on which this logical volume is recorded, whose volume sequence number is identical to the contents of this field.

10.7.2.4 Partition Number (RBP 4)

This field specifies the partition, of the volume specified by the Volume Sequence Number field, identified by the Partition Descriptor whose Partition Number field is identical to the contents of the Partition Number field.

10.7.3 Type 2 Partition Map

This map type identifies a partition in a manner subject to agreement between the originator and recipient of the medium.

It shall be recorded in the format shown in figure 319.

Note 19

Type 2 maps allow partitions to be identified in an implementation specific way that is outside the scope of Part 3. This allows partitions recorded in a manner not specified by Part 3, such as a local disk partition which might locally be referred to as "/dev/dsk/ipi0d2p4" or "a:" or "NODE::DEVICE:", to be part of a logical volume. Type 2 maps may present problems when interchanging media.

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	62	Partition Identifier	bytes

Figure 19 - Type 2 Partition Map format

10.7.3.1 Partition Map Type (RBP 0)

This field shall specify 2.

10.7.3.2 Partition Map Length (RBP 1)

This field shall specify 64.

10.7.3.3 Partition Identifier (RBP 2)

This field shall specify an identification of the partition in a manner subject to agreement between the originator and recipient of the medium.

10.8 Unallocated Space Descriptor

The Unallocated Space Descriptor shall specify extents that are unallocated. It shall be recorded in the format shown in figure 3/20.

BP	Length	Name	Contents
0	16	Descriptor Tag	`tag` (3/7.2) (Tag=7)
16	4	Volume Descriptor SequenceNumber	`Uint32` (1/7.1.5)
20	4	Number of Allocation Descriptors (=N_AD)	`Uint32` (1/7.1.5)
24	N_AD×8	Allocation Descriptors	`extent_ad` (3/7.1)

Figure 20 - Unallocated SpaceDescriptor format

10.8.1 Descriptor Tag (BP 0)

The Tag Identifier field of the `tag` (3/7.2) for this descriptor shall contain 7.

10.8.2 Volume Descriptor Sequence Number (BP 16)

This field shall specify the Volume Descriptor Sequence Number for this descriptor.

10.8.3 Number of Allocation Descriptors (=N_AD) (BP 20)

This field shall specify the number of allocation descriptors recorded in this descriptor.

10.8.4 Allocation Descriptors (BP 24)

This field shall contain N_AD `extent_ad` (3/7.1) descriptors. The Extent Length fields in these descriptors shall be an integral multiple of the logical sectorsize.

Note 20

As specified by 3/8.4.4, the remainder of the last logical sector comprising the Unallocated Space Descriptor is recorded with #00 bytes.

10.9 Terminating Descriptor

A Terminating Descriptor may be recorded to terminate a Volume Descriptor Sequence (see 3/8.4.2). It shall be recorded in the format shown in figure 3/21.

BP	Length	Name	Contents
0	16	Descriptor Tag	`tag` (3/7.2) (Tag=8)
16	496	Reserved	#00 bytes

Figure 21 - Terminating Descriptorformat

10.9.1 Descriptor Tag (BP 0)

The Tag Identifier field of the `tag` (3/7.2) for this descriptor shall contain 8.

10.9.2 Reserved (BP 16)

This field shall be reserved for future standardisation and all bytes shall be set to #00.

10.10 Logical Volume Integrity Descriptor

The Logical Volume Integrity Descriptor shall specify the integrity status of a logical volume and shall be recorded in the format shown in figure 3/22. In the following description, the term "the associated logical volume" shall refer to the logical volume described by the Logical Volume Descriptor specifying the Logical Volume Integrity Sequence in which this descriptor is recorded.

BP	Length	Name	Contents
0	16	Descriptor Tag	`tag` (3/7.2) (Tag=9)
16	12	Recording Date and Time	`timestamp` (1/7.3)
28	4	Integrity Type	`Uint32` (1/7.1.5)
32	8	Next Integrity Extent	`extent_ad` (3/7.1)
40	32	Logical Volume Contents Use	`bytes`
72	4	Number of Partitions (=N_P)	`Uint32` (1/7.1.5)
76	4	Length of Implementation Use(=L_IU)	`Uint32` (1/7.1.5)
80	N_P×4	Free Space Table	`Uint32` (1/7.1.5)
N_P×4+80	N_P×4	Size Table	`Uint32` (1/7.1.5)
N_P×8+80	L_IU	Implementation Use	`bytes`

Figure 22 - Logical Volume Integrity format

10.10.1 Descriptor Tag (BP 0)

The Tag Identifier field of the `tag` (3/7.2) for this descriptor shall contain 9.

10.10.2 Recording Date and Time (BP 16)

This field shall specify the date and time of the day at which this descriptor was recorded.

10.10.3 Integrity Type (BP 28)

This field shall specify the type of Integrity Descriptor. The types are shown in figure 23.

Type	Interpretation
0	Shall mean that the descriptor is an Open Integrity Descriptor
1	Shall mean that the descriptor is a Close Integrity Descriptor
2-255	Reserved for future standardisation

Figure 23 - Logical Volume Integrity type

10.10.4 Next Integrity Extent (BP 32)

This field shall specify the next extent of the Logical Volume Integrity Sequence. The extent shall be within the volume in which this descriptor is recorded within. If the extent's length is 0, no such extent is specified.

10.10.5 Logical Volume Contents Use (BP 40)

This field shall specify information required for interpretation of the information recorded on the associated logical volume. The contents of this field shall be specified by the relevant standard for the interpretation of the information recorded on the associated logical volume.

10.10.6 Number of Partitions (=N_P) (BP 72)

This field shall specify the number of partitions in the associated logical volume

10.10.7 Length of Implementation Use (=L_IU) (BP 76)

This field shall specify the length, in bytes, of the Implementation Use field.

10.10.8 Free Space Table (BP 80)

This field shall contain N_P values, each recorded as a `Uint32`, recorded contiguously starting at the first byte of the field. The *i*th value specifies the amount of available space, in logical blocks, on the partition specified by

the *i*th entry in the Partition Maps field in the associated Logical volume Descriptor. A value of #FFFFFFFF shall mean that the amount of available space is not specified.

10.10.9 Size Table (BP N_P×4+80)

This field shall contain N_P values, each recorded as a `Uint32`, recorded contiguously starting at the first byte of the field. The *i*th value specifies the size, in logical blocks, of the partition specified by the *i*th entry in the Partition Maps field in the associated Logical volume Descriptor. A value of #FFFFFFFF shall mean that the size is not specified.

10.10.10 Implementation Use (BP N_P×8+80)

If L_IU is greater than 0, this field shall specify an identification of an implementation, recorded as a `regid` (1/7.4) in the first 32 bytes of this field, which can recognise and act upon the remainder of this field, which shall be reserved for implementation use and its contents are not specified by this ECMA. The scope of this `regid` shall be the contents of this descriptor, the contents of the associated Logical volume Descriptor, and the contents of the associated logical volume.

11 Levels of medium interchange

Part 3 specifies three levels of medium interchange. The level of a volume shall be that level specifying the most restrictions required to record the volume according to the specifications of Part 3.

In the following level specifications, *N* is the largest logical sector number of a volume.

11.1 Level 1

At level 1, the following restrictions shall apply:

- an Anchor Volume Descriptor Pointer shall be recorded at logical sector 256 and at logical sector N–256
- the Volume Descriptor Sequences specified by the anchor Volume Descriptor Pointer recorded at logical sector 256 and logical sector N–256 shall be recorded in a single extent
- a Logical Volume Descriptor shall contain only Type 1 Partition Maps
- the partitions specified in any Logical Volume Descriptor, catenated together in their recorded order, shall form a single extent
- there shall be exactly one Primary Volume Descriptor in the Main Volume Descriptor Sequence
- there shall be at most one Implementation Use Volume Descriptor in the Main Volume Descriptor Sequence
- there shall be at most one Partition Descriptor with any given Partition Number in the Main Volume Descriptor Sequence
- a volume set shall consist of only one volume.

Note 21

This interchange level provides a simple volume structure.

11.2 Level 2

At level 2, the following restriction shall apply:

- a volume set shall consist of only one volume.

11.3 Level 3

At Level 3, no restrictions shall apply.

Section 3 - Requirements for systems for volume structure

12 Requirements for the description of systems

Part 3 specifies that certain information shall be communicated between a user and an implementation. Each implementation that conforms to Part 3 shall have a description that identifies the means by which the user may supply or obtain such information.

Note 22

The specifics of the description and the means referred to above will vary from implementation to implementation. For example, an implementation might support two interfaces: a preferred, convenient interface which might vet user input, and a deprecated low level interface which allows any input specified by Part 3.

13 Requirements for an originating system

13.1 General

The implementation shall be capable of recording all descriptors specified in 3/10 on a volume set according to one of the medium interchange levels specified in 3/11.

The implementation shall not change the Maximum Interchange Level field nor the Maximum Character Set List field in a Primary Volume Descriptor except when directed to do so by the user.

The implementation shall be capable of recording a list of character sets (see 1/7.2.11) in which the bit for Character Set Type CS2 shall be set to ONE.

If the user specifies a volume without specifying which Primary Volume Descriptor to use, then the implementation shall use the Primary Volume Descriptor having Primary Volume Descriptor Number 0.

The implementation may only alter the Dirty or Protected bits of any `regid` (1/7.4) field of a descriptor specified by Part 3 when directed to do so by the user.

If any information in the scope of a `regid` (1/7.4) is modified and the implementation cannot ensure that the information recorded within the scope of that `regid` still conforms to the agreement implied by the identification in that `regid`, then, when directed to do so by the user, the implementation shall set the Dirty bit of the Flags field of that `regid` to ONE and should not alter the Identifier field of that `regid`.

13.2 Mandatory access by user

13.2.1 Descriptors

The implementation shall allow the user to supply the information that is to be recorded in each of the following descriptor fields, and shall supply the information for a field if the user does not supply it.

Primary Volume Descriptor

- Primary Volume Descriptor Number
- Volume Identifier
- Volume Sequence Number
- Maximum Volume Sequence Number
- Maximum Interchange Level
- Maximum Character Set List
- Volume Set Identifier
- Descriptor Character Set

Partition Descriptor

- Partition Number
- Partition Contents
- Access Type
- Partition Starting Location
- Partition Length

Logical Volume Descriptor

- Descriptor Character Set
- Logical Volume Identifier
- Logical Block Size
- Number of Partition Maps
- Partition Maps

13.3 Optional access by user

If the implementation permits the user to supply the information that is to be recorded in the extents, if any, specified by the Volume Copyright Notice and Volume Abstract fields in a Primary Volume Descriptor, the implementation shall record such information as supplied by the user.

13.3.1 Descriptors

If the implementation permits the user to supply the information that is to be recorded in any of the following descriptor fields, the implementation shall record such information as supplied by the user, and shall supply the information for a field if the user does not supply it.

Primary Volume Descriptor

- Character Set List
- Explanatory Character Set
- Application Identifier

Logical Volume Descriptor

- Domain Identifier

13.3.2 Multivolume volume sets

The implementation shall not be required to record information on the volumes of a volume set that have been assigned a sequence number n, where $1 \le n < m$, after any information has been recorded on the volume of the volume set that has been assigned sequence number m.

The implementation shall not be required to record information on the volume of a volume set that has been assigned sequence number $m+1$ if there is sufficient space to record the information on the volume that has been assigned a sequence number n, where $1 \le n \le m$.

14 Requirements for a receiving system

14.1 General

The implementation shall be capable of interpreting all descriptors as specified in 3/10 on a volume set according to one of the medium interchange levels specified in Part 3.

If the user specifies a volume without specifying which Primary Volume Descriptor to use, then the implementation shall use the Primary Volume Descriptor having Primary Volume Descriptor Number 0.

14.2 Mandatory access by user

14.2.1 Descriptors

The implementation shall make available to the user the information that is recorded in the extent , if any, specified by the Volume Copyright Notice and Volume Abstract fields in a Primary Volume Descriptor.

The implementation shall allow the user to access the information that is recorded in each of the following descriptor fields

Primary Volume Descriptor

- Volume Identifier
- Volume Sequence Number
- Maximum Volume Sequence Number
- Interchange Level
- Maximum Interchange Level
- Maximum Character Set List
- Volume Set Identifier
- Descriptor Character Set
- Explanatory Character Set

Partition Descriptor

- Partition Number
- Partition Contents
- Access Type
- Partition Starting Location
- Partition Length

Logical Volume Descriptor

- Logical Volume Identifier
- Domain Identifier

Standard ECMA - 167

Volume and File Structure of Write-Once and Rewritable Media using Non-Sequential Recording for Information Interchange

Part 4 : File structure

Section 1 - General

1 Scope

Part 4 specifies a format and associated system requirements for file structure by specifying:

- the placement of files;
- the attributes of the files;
- the relationship among files of a logical volume;
- levels of medium interchange;
- requirements for the processes which are provided within information processing systems, to enable information to be interchanged between different systems; for this purpose, it specifies the functions to be provided within systems which are intended to originate or receive media which conform to Part 4.

2 Parts references

See 1/2.

3 Part interface

This clause specifies the interface of Part 4 to other standards or Parts.

3.1 Input

Part 4 requires the specification of the following by another standard or Part.

- Volume sets of one or more volumes.
- A means for assigning volume sequence numbers (see 4/8.1).
- Logical volumes composed of partitions.
- Numeric identification of the partitions within a logical volume.
- If the volume is recorded according to Part 3, the partitions shall be numbered according to 3/8.8.
- Identification of a logical volume on which one or more file sets may be recorded.
- Division of the partitions for a logical volume into fixed size logical blocks.
- Numeric identification of the logical blocks within a partition.
- The size of a logical block for a logical volume. This shall be an integral multiple of 512.
- A means for detecting if a logical block is unrecorded.
- If the volume is recorded according to Part 3, a logical block shall be unrecorded if all of the logical sectors comprising that logical block are unrecorded. A logical block should either be completely recorded or unrecorded.
- A means for identifying the first extent of the File Set Descriptor Sequence (see 4/8.3.1) of a logical volume;
- If the volume is recorded according to Part 3, the extent in which the first File Set Descriptor Sequence of the logical volume is recorded shall be identified by a long_ad (4/14.14.2) recorded in the Logical Volume Contents Use field (see 3/10.6.7) of the Logical Volume Descriptor describing the logical volume in which the File Set Descriptors are recorded.
- A means for specifying the Logical Volume Header Descriptor (see 4/14.15) of a logical volume
- If the volume is recorded according to Part 3, the Logical Volume Header descriptor shall be recorded in the Logical Volume Contents Use field (see 3/10.10.5) of the prevailing Logical Volume Integrity Descriptor for the logical volume

– A means for identifying the following for each partition of the logical volumeon which a file set is recorded:

- Unallocated Space Table and Unallocated Bit Map (see 4/10)
- Freed Space Table and Freed Bit Map (see 4/10)
- Partition Integrity Table (see 4/11)

If the volume is recorded according to Part 3, the Partition Contents Use field (see 3/10.5.6) of the Partition Descriptor (see 3/10.5) describing the partition shall be recorded as a Partition Header Descriptor (see 4/14.3). Such a Partition Descriptor shall have "+NSR02" recorded in the Partition Contents field.

3.2 Output

Part 4 specifies the following which may be used by other standards or Parts.

– Data space of a file (see 4/8.8.2).
– Attributes of a file.
– Attributes of a directory.
– Attributes of a directoryhierarchy.

4 Conformance

See 1/3.

5 Definitions

In addition to the definitions of Part 1 (see 1/5), the following definitions apply for Part 4.

5.1 extent

A set of logical blocks, the logical block numbers of which form a continuous ascending sequence. The address, or location, of an extent is the number of the first logical block in the sequence.

5.2 file set

A collection of files and directories.

5.3 group ID

An identification of a group of users.

5.4 logical block

The unit of allocation of a logical volume.

5.5 logical volume

A nonempty set of partitions over which one or more file sets are recorded.

5.6 partition

An extent of logical blocks within a volume.

5.7 user ID

An identification of a user.

6 Notation

The notation of Part 1 (see 1/7) applies to Part 4.

7 Basic types

In addition to the basic types of Part 1 (see 1/7), the following basic types apply for Part 4.

7.1 Recorded address

A logical block address may be specified by an `lb_addr` recorded in the format shown in figure 4/1.

RBP	Length	Name	Contents
0	4	Logical Block Number	`Uint32` (1/7.1.5)
4	2	Partition Reference Number	`Uint16` (1/7.1.3)

Figure 1 - `lb_addr` format

7.1.1 Logical Block Number (RBP 0)

This field specifies the logical block number relative to the start of the partition identified by the Partition Reference Number field. The value 0 shall refer to the first logical block in the partition.

7.1.2 Partition Reference Number (RBP 4)

This field contains the numeric identification of a partition within a logical volume (see 4/3.1).

7.2 Descriptor Tag

Certain descriptors specified in Part 4 have a 16 byte structure, or `tag`, at the start of the descriptor with the format given in figure 4/2.

Note 1

There are two main motivations for using a general tag structure. The first is that most descriptors need a generic way to handle issues of CRCs and format versions. The second motivation is to support recovery after the medium has been damaged or corrupted in some (unspecified) way. With the tag described here, structures are self identifying and can be verified with very little context.

RBP	Length	Name	Contents
0	2	Tag Identifier	`Uint16` (1/7.1.3)
2	2	Descriptor Version	`Uint16` (1/7.1.3)
4	1	Tag Checksum	`Uint8` (1/7.1.1)
5	1	Reserved	#00 byte
6	2	Tag Serial Number	`Uint16` (1/7.1.3)
8	2	Descriptor CRC	`Uint16` (1/7.1.3)
10	2	Descriptor CRC Length	`Uint16` (1/7.1.3)
12	4	Tag Location	`Uint32` (1/7.1.5)

Figure 2 - `tag` format

7.2.1 Tag Identifier (RBP 0)

This field shall specify an identification of the descriptor type. Type 0 shall specify that the format of this descriptor is not specified by Part 4. Types 1-7 and 9 are specified in Part 3. Type 8 is specified identically in Part 3 and Part 4. Types 256-265 are specified in Part 4. All other types are reserved for future standardisation. The descriptor types specified by Part 4 are shown in figure 4/3.

Type	Interpretation
8	Terminating Descriptor (3/10.9 and 4/14.2)
256	File Set Descriptor (4/14.1)
257	File Identifier Descriptor (4/14.4)
258	Allocation Extent Descriptor (4/14.5)
259	Indirect Entry (4/14.7)
260	Terminal Entry (4/14.8)
261	File Entry (4/14.9)
262	Extended Attribute Header Descriptor (4/14.10.1)
263	Unallocated Space Entry (4/14.11)
264	Space Bitmap Descriptor (4/14.12)
265	Partition Integrity Entry (4/14.13)

Figure 3 - Descriptor interpretation

7.2.2 Descriptor Version (RBP 2)

This field shall specify the version of this descriptor. The value 2 shall indicate the structure of Part 4.

7.2.3 Tag Checksum (RBP 4)

This field shall specify the sum modulo 256 of bytes 0-3 and 5-15 of the `tag`.

7.2.4 Reserved (RBP 5)

This field shall be reserved for future standardisation and shall be set to #00.

7.2.5 Tag Serial Number (RBP 6)

This field shall specify an identification of a set of descriptors. If the field contains 0, then no such identification is specified.

Note 2

This field can be used to distinguish between groups of descriptors. For example, when reusing rewritable media, an implementation might choose a different serial number from the previous use when initialising a volume. Thus, a disaster recovery mechanism can avoid recovering prior and unintended data. The only alternative to this scheme would be to force volume initialisation to clear the volume.

7.2.6 Descriptor CRC (RBP 8)

This field shall specify the CRC of the bytes of the descriptor starting at the first byte after the descriptor tag . The number of bytes shall be specified by the Descriptor CRC Length field. The CRC shall be 16 bits long and be generated by the CRC-ITU-T polynomial (see ITU-T V.41):

$$x^{16} + x^{12} + x^5 + 1$$

Note 3

As an example, the CRC of the three bytes #70 #6A #77 is #3299. Implementations can avoid calculating the CRC by setting the Descriptor CRC Length to 0, as then the Descriptor CRC shall be 0.

7.2.7 Descriptor CRC Length (RBP 10)

This field specifies how many bytes were used in calculating the Descriptor CRC

7.2.8 Tag Location (RBP 12)

This field shall specify the number of the logical block , within the partition the descriptor is recorded on, containing the first byte of the descriptor.

Note 4

The location of the tag may appear to be redundant but its primary purpose is to make it extremely likely that if the first 16 bytes of a logical sectoror logical block is a consistent descriptor tag, then it is a descriptor tag.

Section 2 - Requirements for the medium for file structure

8 File structure

8.1 Volume set

Each volume in a volume setshall have an assigned volume sequence number as specified in 4/3.1.

8.2 Arrangement of information on a volume set

A logical volume and its related file sets shall be recorded on a volume set. Identification of the logical volumes (see 4/3.1) and related File Set Descriptors for all the logical volumes in a volume set shall be recorded in the volume with the highest volume sequence number in the volume set.

8.3 Arrangement of information on a logical volume

Part 4 takes a logical volume to be a set of partitions on one or more volumes. Each partition shall be considered as an extent of logical blocks, and shall have an identification as specified in the input parameters for Part 4 (see 4/3.1). An address within a logical volume has two parts; one part identifies a partition within the logical volume and the other part specifies a logical block numberrelative to the start of that partition.

8.3.1 File Set Descriptor Sequence

A File Set Descriptor Sequence shall be recorded as a sequence of extents within a logical volume . An extent of the File Set Descriptor Sequence shall be recorded according to the schemshown in figure 4/4.

```
[File Set Descriptor Sequence extent]{
            <File Set Descriptor >0+
 [Terminator]{
        <File Set Descriptor >
       | <Terminating Descriptor >
       | <unrecorded logical block >
 } <trailing logical block >0+
} 0+
```

Figure 4 - File set descriptor sequenceschema

The first extent of the sequence shall be identified by the input parameters (see 4/3.1) of Part 4. Each, if any, subsequent extent of the sequence shall be identified by the Next Extent field of a File Set Descriptor. An extent of the sequence shall be terminated by either an unrecorded logical block (see 4/3.1), a Terminating Descriptor (see 4/14.2), or by a File Set Descriptor whose Next Extent field identifies a subsequent extent of the sequence.

All File Set Descriptors shall have an assigned file set descriptor number. All File Set Descriptors with identical file set descriptor numbers shall have identical contents.

All file sets shall have an assigned file set number. Of the File Set Descriptors of a File Set Descriptor Sequence with identical file set numbers, the one with the highest file set descriptor number shall be used. This instance shall be referred to as the prevailing instance.

One of the File Set Descriptors of a File Set Descriptor Sequence shall have a file set number of 0.

A File Set Descriptor shall specify a file set identification. No prevailing instance of a File Set Descriptor shall specify the same file set identification as any other prevailing instance of a File Set Descriptor.

8.4 Arrangement of information on a partition

A means for identifying the location of the following for each partition of the logical volume on which a file set is recorded shall be specified by the inputparameters (see 4/3.1) of Part 4.

- Unallocated SpaceTable and Unallocated Bit Map (see 4/10)

- Freed Space Table and Freed Bit Map (see 4/10)
- Partition Integrity Table (see 4/11)

8.5 File set

A file set shall be identified by a File Set Descriptor which identifies the root of a directory hierarchy (see 4/8.6) describing a set of files and certain attributes of the file set. A prevailing File Set Descriptor specifies

- the name of the logical volume it is recorded on
- the set of characters allowed in certain fields of descriptors associated with the file set identified by the File Set Descriptor
- an identification of the root of the directory hierarchy describing the files of the file set identified by the File Set Descriptor
- copyright and abstract information for the file set.

8.6 Directories

A directory contains zero or more file or directory identifications. A directory hierarchy shall be a set of directories descended from a single root directory

A directory shall contain a set of directory descriptors, each of which identifies a parent directory or a component file or a component subdirectory. A directory descriptor that identifies a parent directory or a component file or subdirectory by specifying the address of an ICB (see 4/8.10.1) for that component shall be recorded as a File Identifier Descriptor (see 4/14.4). A directory descriptor that identifies a component file or subdirectory of the directory by specifying the pathname of the actual file or directory shall be referred to as an alias and shall be recorded as a File Identifier Descriptor specifying a file whose type is a symbolic link (see 4/14.6.6).

A directory identifying another directory by other than an alias shall be called a parent directory of the identified directory. The identified directory shall be called a subdirectory of the parent directory. Different directories may have the same parent directory. A directory shall have only one parent directory. The parent directory of the root directory shall be the root directory.

Each directory descriptor shall specify the name of a component file or the name of a component subdirectory, or identify the parent directory of the directory. The length, in bytes, of the name of a component file or subdirectory shall be greater than 0. Each directory descriptor shall contain an indication of whether the identified component is a directory. When the descriptor identifies an alias, this indication is contained in the directory descriptor for the file or directory identified by the pathname specified by the alias.

A directory shall be recorded according to the schema shown in figure 45.

```
{
          <File Identifier Descriptor>
} 0+
```

Figure 5 - Directory schema

For the descriptors in a directory

- there shall not be more than one descriptor with the same File Identifier (see 4/14.4.8) and File Version Number (see 4/14.4.2).
- a descriptor identifying a directory shall have a File Version Number of 1.
- there shall be exactly one File Identifier Descriptor identifying the parent directory (see 4/14.4.3).
- the descriptors shall be ordered according to 4/8.6.1 and 4/14.6.8.

A File Entry specifying a file in which a directory is recorded shall not specify a Character Set Information Extended Attribute.

Note5

The character set specifying the d-characters (1/7.2) used in the directory descriptors is specified in the File Set Descriptor for the directory hierarchyof which the directory is a member.

8.6.1 Order of directory descriptors

If the directory descriptors of a directory are sorted according to Part 4, they shall be ordered by the following criteria in descending order of significance:

1. In ascending order according to the relative value of File Identifier, where File Identifiers shall be valued as follows:

   - If the File Identifiers being compared have the same value in all byte positions, the File Identifiers shall be equal in value.

   - If the File Identifiers being compared do not contain the same number of byte positions, the File Identifiers shall be treated as if they are of equal length by padding the shorter File Identifier on the right with #00 bytes. After any such padding, the File Identifiers shall be compared one byte at a time, in ascending byte position order, until a byte position is found that does not contain the same value in both File Identifiers. The File Identifier with the greater byte value, comparing values of the bytes as unsigned integers, shall be considered greater.

2. In descending order according to the relative value of File Version Number.

Note 6

Sorting applies to files and aliases having been marked as deleted in the File Characteristics field. The order is independent of the `charspec` *(1/7.2.1) applying to the directory because the File Identifiers are sorted as if they were binary values.*

8.6.2 Directory hierarchy size restrictions

The sum of the number of directories and the number of files described by the directories of a directory hierarchy shall be less than 2^{32}.

8.7 Pathname

A pathname may be used to specify a file or directory by name. The length, in bytes, of this pathname shall be greater than 0. The pathname shall consist of a sequence of one or more path components (see 4/14.16) as follows:

- Unless otherwise specified, a component shall be interpreted relative to the directory specified by its predecessor. The predecessor of the initial component shall be the directory in which the pathname is described.
- The final component shall specify either a directory, or a file, or an alias which resolves to either a directory or file.
- Each other component shall specify either a directory or an alias which resolves to a directory.

8.7.1 Resolved pathname

Within a directory hierarchy, every pathname specifying a file or directory has an equivalent resolved pathname. A resolved pathname is a pathname where

- the first component is a Path Componentwith a Component Type of 2.
- each other component is a Path Componentwith a Component Type of 5 and is not an alias

The length of a resolved pathnameshall be the sum of the following:

- the value of the Component Identifier Length field for each component;
- the number of components

Note 7

The resolved pathname is mainly used in 4/15. Note that the length defined here corresponds to that of a theoretical pathname, rather than a pathname that an implementation might use. In particular, it assumes that the component separator is one byte long, which is typically true but is false for certain character set

Note that the length of the resolved pathname does not provide for the length of the file version number associated with the final component of the pathname.

8.8 Files

A file shall be described by a File Entry (see 4/14.9) which shall specify the attributes of the file and the location of the file's recorded data. The data of a file shall be recorded in either of the following:

– An ordered sequence of extents of logical blocks (see `short_ad` (4/14.14.1), long_ad (4/14.14.2) and ext_ad (4/14.14.3). The extents may be recorded or unrecorded, and allocated or unallocated. The extents, if specified as long_ad (4/14.14.2) or ext_ad (4/14.14.3), may be located on different partitions which may be on different volumes.

– The Allocation Descriptors field of a File Entry.

Except where specified in Part 4, neither the interpretation of the information in a file nor the structure of the information in a file is specified by Part 4.

8.8.1 Attributes of a file

The File Entry specifies the attributes of a file. Some of the attributes shall be recorded in fields in the File Entry itself; the remainder shall be recorded as extended attributes. Extended attributes shall be recorded in extended attributes spaces as described in 4/9. The attributes of a file specified by this Part are recorded in extended attributes, and in the following fields of a File Entry and of the `icbtag` (4/14.6) recorded as the contents of the ICB Tag field of the File Entry.

`icbtag`

– Strategy Type
– Strategy Parameter
– File Type
– Flags

File Entry

– Uid
– Gid
– Permissions
– File Link Count
– Record Format
– Record Display Attributes
– Record Length
– Information Length
– Logical Blocks Recorded
– Access Date and Time
– Modification Date and Time
– Attribute Date and Time
– Checkpoint
– Implementation Identifier

Note 8

The information in the File Identifier Descriptor (see 4/14.4.3) for a file pertains only to the identification of the file and is not considered an attribute of the file.

8.8.2 Data space of a file

The data space of a file shall be the following:

- If the file is recorded as the contents of an ordered sequence of extents of logical block s, these extents shall be the data space of the file. The bytes in the data space shall be numbered with consecutive integers assigned in an ascending sequence. The numbering shall start from 0 which shall be assigned to the first byte of the first logical block of the first extent, if any, of the data space.
- If the file is recorded in the Allocation Descriptor s field of a File Entry, the number of bytes specified by the Length of Allocation Descriptors field of the File Entry starting with the first byte of the Allocation Descriptors field of the File Entry shall be the data space of the file. The bytes in the data space shall be numbered with consecutive integers assigned in an ascending sequence. The numbering shall start from 0 which shall be assigned to the first byte, if any, of the data space.

The number of bytes in the data space of a file shall be referred to as the information length of the file (see 4/14.9.10).

Note 9

Some record formats (see 4/14.9.7) specify records of sequences of characters delimited by a specific character sequence. If detection of these delimiter sequences requires knowledge of the character set encoding, such as would be the case if code extension characters (see 1/7.2.9.1) are used, then a Character Set Information extended attribute should be recorded.

8.9 Record structure

The information in a file may be organised as a set of records according to 4/14.9.7. The structure and attributes of these records are specified in Part 5. For the purposes of Part 5, the data space of a file is specified by 4/8.8.2 and

- if the Character Set Information extended attribute is specified for the file, then that extended attribute specifies how the bytes of the file shall be interpreted as characters,
- if the Character Set Information extended attribute is not specified for the file, then each byte of the file shall be interpreted as a single character, and a byte containing #0A shall be a LINE FEED character, a byte containing #0B shall be a VERTICAL TABULATION character, a byte containing #0C shall be a FORM FEED character, and a byte containing #0D shall be a CARRIAGE RETURN character. These interpretations shall only apply for the purposes of partitioning the file's contents into records, and need not apply to the contents of those records.

Note 10

Some record formats (see Part 5) specify records delimited by a specific character sewquence.

8.10 Information Control Block (ICB)

Each recorded instance of a file shall be described by an entry in an Information Control Block (ICB). The set of entries describing the recorded instances of a file shall be described by entries in one or more ICBs. These ICBs shall form an ICB hierarchy as described in 4/8.10.1.

An ICB shall be a sequence of ICB entries recorded in an extent of logical blocks. The address or location of an ICB shall be the address of the extent. An entry of the sequence shall be one of the following:

- a direct entry, describing a recorded occurrence of a file or a set of extents
- an Indirect Entry (4/14.7), describing another ICB
- a Terminal Entry (4/14.8), indicating that there are no more entries recorded after this entry
- an unrecorded logical block, indicating that there are no more entries recorded after this entry

Each entry, other than an unrecorded logical block entry, shall specify:

- The maximum number of entries that may be recorded in the ICB in which the entry is recorded.
- The number of direct entries recorded in the ICB hierarchy prior to recording the entry.

An ICB entry shall not be recorded until all entries in that ICB with lower addresses have been recorded.

Note 11

Recording an indirect entry does not imply that the ICB specified by that indirect entry is necessarily completely recorded.

The ICB may specify the strategy for building an ICB hierarchy (see 4/14.6.2). Annex 4/ A specifies certain strategies; other strategies may be subject to agreement between the originator and recipient of the medium (see annex 4/A).

Note 12

Part 4 requires a data structure that describes sequences of bytes recorded in a volume. This can be used to record a user's file, the contents of a directory or various system data. Some media, such as write-once optical disks, cannot rerecord a sector once it has been written, and thus Part 4 requires a data structure that can describe successive versions of regions of bytes recorded in a volume. Note that this structure is efficient on rewritable media by simply making the ICB a single direct entry. Alternatively, the same structures allow rewritable media to support a history of all versions of a file.

Whereas there is a single algorithm for traversing an ICB hierarchy, there are many algorithms or strategies for building these hierarchies.

8.10.1 ICB hierarchy

An ICB hierarchy shall be a set of ICBs descended from a root ICB . The root ICB shall be the only ICB at level 0 of an ICB hierarchy. An ICB identifying another ICB by an indirect entry shall be called a parent ICB of the identified ICB. The parent ICB shall be at level m of the ICB hierarchy and the identified ICB shall be at level m+1 of the ICB hierarchy.

Different ICBs may have the same parent ICB.

9 Extended attributes

An extended attribute shall specify an attribute type, an attribute subtype, and may specify attribute specific information. Extended attributes are associated with a file. All the extended attributes associated with a file shall be recorded in one or more extended attributes spaces associated with that file. The term “instances of an extended attribute” shall refer to all extended attributes recorded in the extended attributes space of the file with identical contents of their Attribute Typeand Attribute Subtypefields (see 4/14.10.2).

An attribute type shall be an integer x where $0 \leq x < 2^{32}$.

An attribute subtype shall be an integer x where $0 \leq x < 2^8$.

The attribute types are divided into three classes as follows:

- Attribute types 1, 3, 5, 6, 12, 2 048, and 65 536 are registered according to ISO/IEC 13800 and are recorded as specified in 4/14.10. Attribute types 2, 4, 7, 8, 9, 10 and 11 are registered according to ISO/IEC 13800 and shall not be recorded in the extended attributes space of a file. Attribute types 13 to 2 047 inclusive are reserved for reserved for registration according to ISO/IEC 13800. Attribute type 0 is reserved for future standardisation by ISO/IEC 13800.
- Attribute types 2 049 to 65 535 inclusive shall be registered according to ISO/IEC 13800, are recorded according to 4/14.10.2 and are reserved for implementation use according to ISO/IEC 13800.
- Attribute types 65 537 and above shall be registered according to ISO/IEC 13800, are recorded according to 4/14.10.2 and are reserved for application use according to ISO/IEC 13800.

There shall be

- zero or one instance of each attribute with type 1, 3, 5, 6 or 12.

– zero instances of each attribute with type 0, 2, 4, 7, 8, 9, 10 and 11.

– zero or more instances of each attribute with type 2048 or 65 536.

– zero or more instances of each attribute with type 13 to 2047 inclusive, 2 049 to 65 535 inclusive or greater than 65 536 as specified by the registration according to ISO/IEC 13800.

The interpretation of attribute specific fields for each attribute with type

– 2 048: is specified by the `regid` (1/7.4) recorded in the Implementation Identifier field of the attribute.

– 65 536: is specified by the `regid` (1/7.4) recorded in the Application Identifier field of the attribute.

– 13 to 2 047 inclusive or 2 049 through 65 535 inclusive or greater than 65 536: is specified by the registration according to ISO/IEC 13800 of the attribute type and subtype.

If allowed by the registration of the attribute type, multiple instances of an extended attribute need not be identical; they may have different attribute specific information.

An extended attributes space of a file is one of the following:

– The Extended Attributes field of the file's File Entry.
– A file described by an ICB identified in the file's File Entry.

In each case, an extended attributes space shall be recorded according to the schema shown in figure 4/6.

```
<Extended Attribute  Header Descriptor >
<Extended Attribute > 0+
```

Figure 6 - Extended attributes space schema

Extended attributes shall be recorded contiguously in three nonoverlapping areas within an extended attribute s space as follows:

– The first area, starting with the first byte after the Extended Attribute Header Descriptor, is reserved for the recording of attributes defined in clauses 4/14.10.3 to 4/14.10.7.

– The second area, starting at a byte of the extended attribute s space specified in the Extended Attribute Header descriptor, is reserved for the recording of attribute types 2 048 to 65 535 (see 4/14.10.8).

– The third area, starting at a byte of the extended attribute s space specified in the Extended Attribute Header descriptor, is reserved for the recording of attribute types 65 536 and above (see 4/14.10.9).

The following extended attributes are defined in 4/14.10:

– Character Set Information

– File Times

  - File Creation Date and Time, File Deletion Date and Time, File Effective Date and Time, File Last Backup Date and Time

– Information Times

  - Information Creation Date and Time, Information Last Modification Date and Time, Information Expiration Date and Time, Information Effective Date and Time

– Alternate Permissions

– Device Specification

– Application Use

– Implementation Use

Note 13

There need not be any extended attributes associated with a file. The multiple occurrences of attributes of types 2 048 and 65 536, if any, are intended to be distinguished by the contents of their Implementation Identifier and Application Identifier fields respectively. Such occurrences might have differing contents in some attribute specific fields, such as the Implementation Use or Application Use fields.

10 Partition space management

A partition has two types of space managed by Part 4; space ready for allocation (unallocated space), and space that may require preparation before allocation (freed space). In both cases, partition space is specified by a space set which specifies a collection of logical blocks in the partition. A space set shall be recorded as either a Space Table or as a Space Bitmap as specified in 4/10.1.

The Unallocated Space Set of a partition is a space set. If a logical block is in the Unallocated Space Set, it may be allocated for recording.

The Freed Space Set of a partition is a space set. If a logical block is in the Freed Space Set, it may be allocated for recording but may require preparation before recording as specified by the standard for recording.

The Unallocated and Freed SpaceSets shall be identified by the Partition Header Descriptor (4/14.3).

10.1 Space sets

A space set shall be recorded as either a Space Table or as a Space Bitmap.

A Space Table shall be recorded as an ICB hierarchy consisting of indirect entries and unallocated space entries (see 4/14.11). The logical blocks in the space set are all the logical blocks which belong to the extents specified by the last Space Entry in the Space Table.

A Space Bitmap shall be recorded as a Space Bitmap Descriptor which includes a sequence of n bits recorded in a single extent, where n is the number of logical blocks in the partition. The value of bit s is recorded at bit $rem(s,8)$ of byte $ip(s/8)$, where byte 0 is the first byte of the extent. The space set consists of all logical blocks s such that bit s is ONE.

Note 14

It is preferable that the Standard support just one type of space set. However, it is anticipated that the unallocated partition space will be updated fairly often, and that the unallocated partition space will get fragmented. Bitmaps handle the latter case efficiently but are too expensive for the former case on write-once media. In this case, we need the equivalent of an ICB which essentially records many instances of an arbitrary sequence of bytes. In fact, a Space Table is simply an ICB with direct entries specialised for recording extents of space. It is expected, but not required, that rewritable media will use space bitmaps and write-once media will use space tables.

Rewritable media may also require a second space set for logical blocks which may need to be preconditioned before recording. Some rewritable magneto-optic technology requires sectors be cleared before recording and the freed space list allows this clearing to be done asynchronously with the freeing of that space. Clearing large numbers of sectors at one time may also allow use of special hardware features such as clearing a track in one operation. For rewritable media that requires no special preprocessing for rewriting sectors, it is likely that the freed space map will be empty.

11 Partition integrity

Partition integrity specifies the status of the information recorded on the medium and shall be recorded as a Partition Integrity Table specified by the Partition Header Descriptor (see 4/14.3). This is an ICB consisting of Indirect Entries and Partition Integrity Entries (see 4/14.13) as follows:

– An Open Integrity Entry shall be recorded before any data is recorded in the partition since the last Close Integrity Entry, if any, was recorded.

– A Close Integrity Entry may be recorded only after all user data has been completely recorded and the descriptors recorded on the partition conform to Part 4.

– A Stable Integrity Entry may be recorded after all descriptors recorded on the partition conform to Part 4. However, the data in files with a 0, 5, 6, 7 or 9 in the File Type field (see 4/14.6.6) of the File Entry describing the file need not have been recorded.

Note 15

The partition integrity entries provide a standard, portable and convenient way for implementations to indicate that a modified partition has had both its data and control structures properly updated. Because of optical media's large size and relatively slow access, it is particularly important to avoid unnecessary consistency checks over a partition or volume.

As an example, consider a partition mounted as a file system on a computer. When the first write request for that partition is issued, an Open Integrity Entry is recorded prior to performing the write request. When the partition is unmounted, a Close Integrity Entry is recorded after any queued write requests have been performed. Periodically, Stable Integrity Entries may be recorded after bringing up to date the data structures specified in Part 4. Finally, systems, such as file servers, that keep file systems mounted for long periods of time and wish to minimise the risk of a system failure and the use of lengthy recovery procedures, might adopt heuristics such as recording a Close Integrity Entry periodically or after some delay after the last write request.

12 Allocation descriptors

Allocation descriptors (see 4/14.14) specify the location, length, and type of an extent. The extent's type is one of

– recorded data,
– allocated and unrecorded space,
– space neither allocated nor recorded.

The contents of an unallocated or unrecorded extent shall be interpreted as all #00 bytes. Allocation refers to the reservation of one or more extents for current or future use, guaranteeing its availability for recording and making it unavailable for any other purpose.

A sequence of allocation descriptors shall be recorded contiguously within a field or an extent

A field or an extent of a sequence of allocation descriptors shall be terminated by one of

– the end of the field,

– an allocation descriptor whose Extent Length field is 0,

– an allocation descriptor identifying a continuation extent in which the recording of the sequence of allocation descriptors is continued. The continuation extent shall be recorded according to the schemashown in figure 4/7.

```
[Extent of Allocation Descriptor s]{
     <Allocation Extent Descriptor >
     <allocation descriptor> 1+
}
```

Figure 7 - Continuation extentschema

Allocation descriptors have an associated Information Length, which is the amount of information, in bytes, in the extent. The Extended Allocation Descriptor, or `ext_ad` (4/14.14.3), specifies the Information Length as a field; for all other allocation descriptors, the Information Length shall be the same as the length of the extent.

12.1 Description of Files

The sequence of the allocation descriptors describing the extents of a file shall be recorded as a File Body followed by a File Tail according to the schemashown in figure 4/8.

```
[File Body]{
      <allocation descriptor>(extent  length is a multiple of LBS) 0+
      <allocation descriptor> 0+1
}
[File Tail]{
      <allocation descriptor>(unrecorded and allocated) 0+
}
```

Figure 8 - File extents schema

LBS denotes the logical block size. The type of allocation descriptor shall be specified by the Flags field in the ICB Tag field (see 4/14.6.8).

Note 16

A sparse file, such as a large file with data recorded only at the beginning and the end of the file, might be recorded as two allocated and recorded extents separated by an unallocated and unrecorded extent

13 Recording of descriptors

All the descriptors in Part 4 whose format is specified with Byte Positions (BP) shall be recorded so that the first byte of the descriptor coincides with the first byte of a logical block . All the descriptors in Part 4 whose format is specified with Byte Positions (BP), except for the Space Bitmap Descriptor, shall have a length no larger than the size of a logical block.

The descriptors in Part 4 whose format is specified with Relative Byte Positions (RBP) have no restrictions on where they may be recorded within a logical block , except that their location within a descriptor shall be specified in the description of the applicable descriptor.

When the descriptors described in Part 4 are recorded in a logical block, all space, if any, after the end of the last descriptor up to the end of the logical block is reserved for future standardisation and shall be recorded as all #00 bytes.

14 File Data Structures

14.1 File Set Descriptor

The File Set Descriptor shall identify a set of files and directories and shall be recorded in the format shown in figure 4/9.

BP	Length	Name	Contents
0	16	Descriptor Tag	`tag` (4/7.2)(Tag=256)
16	12	Recording Date and Time	`timestamp` (1/7.3)
28	2	Interchange Level	`Uint16` (1/7.1.3)
30	2	Maximum Interchange Level	`Uint16` (1/7.1.3)
32	4	Character Set List	`Uint32` (1/7.1.5)
36	4	Maximum Character Set List	`Uint32` (1/7.1.5)
40	4	File Set Number	`Uint32` (1/7.1.5)
44	4	File Set Descriptor Number	`Uint32` (1/7.1.5)
48	64	Logical Volume Identifier Character Set	`charspec` (1/7.2.1)
112	128	Logical Volume Identifier	`dstring` (1/7.2.12)
240	64	File Set Character Set	`charspec` (1/7.2.1)
304	32	File Set Identifier	`dstring` (1/7.2.12)
336	32	Copyright File Identifier	`dstring` (1/7.2.12)
368	32	Abstract File Identifier	`dstring` (1/7.2.12)
400	16	Root DirectoryICB	`long_ad` (4/14.14.2)
416	32	Domain Identifier	`regid` (1/7.4)
448	16	Next Extent	`long_ad` (4/14.14.2)
464	48	Reserved	#00 bytes

Figure 9 - File Set Descriptor format

14.1.1 Descriptor Tag (BP 0)

The Tag Identifier field of the `tag` (4/7.2) for this descriptor shall contain 256.

14.1.2 Recording Date and Time (BP 16)

This field shall specify the date and time of the day at which this descriptor was recorded.

14.1.3 Interchange Level (BP 28)

This field shall specify the current level of medium interchange (4/15) of the file set described by this descriptor.

14.1.4 Maximum Interchange Level (BP 30)

This field shall specify the maximum value that may be specified for the Interchange Level field of this descriptor.

14.1.5 Character Set List (BP 32)

This field shall identify the character sets specified by any field, whose contents are specified to be a `charspec` (1/7.2.1), of any descriptor specified in Part 4 and recorded in the file set described by this descriptor.

14.1.7 Maximum Character Set List (BP 36)

The Character Set List field in this descriptor shall not specify a character set (see 1/7.2.11) not specified by the Maximum Character Set List field.

Note 17

The Interchange Level, Maximum Interchange Level, Character Set List and Maximum Character Set List fields permit an implementation to:

- *determine whether it can process all of the information in the file set*
- *restrict the recording of information in the file set so that the file set does not exceed the level given in the Maximum Interchange Level field*

- *restrict the recording of information in the file set so that all character sets recorded belong to the Maximum Character Set List field.*

This allows a user to create a file set that can be processed when it is returned to the user.

14.1.7 File Set Number (BP 40)

This field shall specify the assigned file set number for this descriptor.

14.1.8 File Set Descriptor Number (BP 44)

This field shall specify the assigned file set descriptor number for this descriptor.

14.1.9 Logical Volume Identifier Character Set (BP 48)

This field shall specify the d-characters (1/7.2) allowed in the Logical Volume Identifier field.

If the volume is recorded according to Part 3, the contents of this field shall be identical to the contents of the Descriptor Character Set field of the Logical Volume Descriptor describing the logical volume on which the file set described by this File Set Descriptor is recorded.

14.1.10 Logical Volume Identifier (BP 112)

This field shall specify an identification of the logical volume on which the file set is recorded.

If the volume is recorded according to Part 3, the contents of this field shall be identical to the contents of the Logical Volume Identifier field of the Logical Volume Descriptor describing the logical volume on which the file set described by this File Set Descriptor is recorded.

14.1.11 File Set Character Set (BP 240)

This field shall specify the d-characters (1/7.2) allowed in certain fields of descriptors specified by Part 4 which have been specified as containing d-characters.

Note 18

Part 4 does not specify the relationship between the contents of the File Set Character Set field and the Logical Volume Identifier Character Set fields or the relationship of those fields to any other fields specified by Part 4 or by another standard.

14.1.12 File Set Identifier (BP 304)

This field shall specify an identification of the file set described by this File Set Descriptor.

14.1.13 Copyright File Identifier (BP 336)

This field shall identify a file in the root directory containing a copyright statement for the information recorded in the file set identified by this File Set Descriptor. If the field contains all #00 bytes, then no such file is identified.

14.1.14 Abstract File Identifier (BP 368)

This field shall identify a file in the root directory containing an abstract for the information recorded in the file set identified by this File Set Descriptor. If the field contains all #00 bytes, then no such file is identified.

14.1.15 Root Directory ICB (BP 400)

This field shall specify the location of an ICB describing the root directory of the directory hierarchy associated with the file set identified by this File Set Descriptor If the extent's length is 0, no such ICB is specified.

14.1.16 Domain Identifier (BP 416)

This field shall specify an identification of a domain which shall specify rules on the use of, and restrictions on, certain fields in descriptors subject to agreement between the originator and recipient of the medium. If this field contains all #00 bytes, then no such domain is identified. The scope of this `regid` (1/7.4) shall include all information recorded in the file set described by this descriptor.

14.1.17 Next Extent (BP 448)

This field shall specify the next extent where File Set Descriptors may be recorded. If the extent's length is 0, no such extent is specified.

14.1.18 Reserved (BP 464)

This field shall be reserved for future standardisation and all bytes shall be set to #00.

14.2 Terminating Descriptor

A Terminating Descriptor may be recorded to terminate an extent of a File Set Descriptor Sequence (see 4/8.3.1). It shall be recorded in the format shown in figure 410.

BP	Length	Name	Contents
0	16	Descriptor Tag	`tag` (3/7.2) (Tag=8)
16	496	Reserved	#00 bytes

Figure 10 - Terminating Descriptorformat

14.2.1 Descriptor Tag (BP 0)

The Tag Identifier field of the `tag` (3/7.2) for this descriptor shall contain 8.

14.2.2 Reserved (BP 16)

This field shall be reserved for future standardisation and all bytes shall be set to #00.

14.3 Partition Header Descriptor

The Partition Header Descriptor shall specify the Unallocated Space Set, the Freed Space Set, and the Partition Integrity Table. As specified in 4/3.1, it shall be recorded with the format shown in figure 411.

RBP	Length	Name	Contents
0	8	Unallocated SpaceTable	`short_ad` (4/14.14.1)
8	8	Unallocated SpaceBitmap	`short_ad` (4/14.14.1)
16	8	Partition Integrity Table	`short_ad` (4/14.14.1)
24	8	Freed Space Table	`short_ad` (4/14.14.1)
32	8	Freed Space Bitmap	`short_ad` (4/14.14.1)
40	88	Reserved	#00 bytes

Figure 11 - Partition Header Descriptor format

14.3.1 Unallocated Space Table (RBP 0)

This field shall specify the Unallocated Space Table for this partition (see 4/10). If the extent's length is 0, then no Unallocated Space Table is specified.

14.3.2 Unallocated Space Bitmap (RBP 8)

This field shall identify the extent in which the Unallocated Space Bitmap for this partition (see 4/10) is recorded. If the extent's length is 0, then no Unallocated Space Bitmap is specified.

14.3.3 Partition Integrity Table (RBP 16)

This field shall specify the Partition Integrity Table for this partition (see 4/11). If the extent 's length is 0, then no Partition Integrity Table is specified.

14.3.4 Freed Space Table (RBP 24)

This field shall specify the Freed Space Table for this partition (see 4/10). If the extent 's length is 0, then no Freed Space Table is specified.

14.3.5 Freed Space Bitmap (RBP 32)

This field shall identify the extent in which the Freed Space Bitmap for this partition (see 4/10) is recorded. If the extent's length is 0, then no Freed Space Bitmap is specified.

14.3.6 Reserved (RBP 40)

This field shall be reserved for future standardisation and all bytes shall be set to #00.

14.4 File Identifier Descriptor

A File Identifier Descriptor shall be recorded in the format shown in figure 4/12.

RBP	Length	Name	Contents
0	16	Descriptor Tag	`tag` (4/7.2)(Tag=257)
16	2	File Version Number	`Uint16` (1/7.1.3)
18	1	File Characteristics	`Uint8` (1/7.1.1)
19	1	Length of File Identifier (=L_FI)	`Uint8` (1/7.1.1)
20	16	ICB	`long_ad` (4/14.14.2)
36	2	Length of Implementation Use(=L_IU)	`Uint16` (1/7.1.3)
38	L_IU	Implementation Use	bytes
[L_IU+38]	L_FI	File Identifier	d-characters (1/7.2)
[L_FI+L_IU+38]	*	Padding	bytes

Figure 12 - File Identifier Descriptor format

14.4.1 Descriptor Tag (RBP 0)

The Tag Identifier field of the `tag` (4/7.2) for this descriptor shall contain 257.

14.4.2 File Version Number (RBP 16)

This field shall specify the file version number of the file specified by the File Identifier field as a number in the range 1 to 32 767 inclusive. The numbers 32768 to 65 535 inclusive are reserved for future standardisation.

14.4.3 File Characteristics (RBP 18)

This field shall specify certain characteristics of the file as shown in figure 4/13.

Bit	Interpretation
0	Existence: If set to ZERO, shall mean that the existence of the file shall be made known to the user; If set to ONE, shall mean that the existence of the file need not be made known to the user.
1	Directory: If set to ZERO, shall mean that the file is not a directory (see 4/14.6.6); If set to ONE, shall mean that the file is a directory. *Note 1 - If the file is a symbolic link (see 4/14.6.6), the Directory bit is set to 0.*
2	Deleted: If set to ONE, shall mean this File Identifier Descriptor identifies a file that has been deleted; If set to ZERO, shall mean that this File Identifier Descriptor identifies a file that has not been deleted. *Note 2 - The Deleted bit allows a file to be deleted from a directory by only rewriting the logical block (s) containing the File Identifier Descriptor. Note that even if the Deleted bit is set to ONE, all the descriptor's fields still need to be valid.*
3	Parent: If set to ONE, shall mean that the ICB field of this descriptor identifies the ICB associated with the file in which is recorded the parent directory of the directory that this descriptor is recorded in; If set to ZERO, shall mean that the ICB field identifies the ICB associated with the file specified by this descriptor
4-7	Shall be reserved for future standardisation and all bits shall be set to ZERO.

Figure 13 - File characteristics

If the Parent bit is set to ONE, then the Directorybit shall be set to ONE.

14.4.4 Length of File Identifier (=L_FI) (RBP 19)

This field shall specify the length, in bytes, of the File Identifier field. If the Parent bit of the File Characteristics field is set to ONE, the length of the File Identifier field shall be 0.

14.4.5 ICB (RBP 20)

This field shall specify the address of an ICB describing the file. If the Delete bit of the File Characteristics field of the File Identifier Descriptor is set to ONE, the ICB field may contain all #00 bytes, in which case no ICB is specified.

14.4.6 Length of Implementation Use (=L_IU) (RBP 36)

This field shall specify the length, in bytes, of the Implementation Use field. L_IU shall be an integral multiple of 4.

14.4.7 Implementation Use (RBP 38)

If L_IU is greater than 0, this field shall specify an identification of an implementation, recorded as a `regid` (1/7.4) in the first 32 bytes of this field, which can recognise and act upon the remainder of this field, which shall be reserved for implementation use and its contents are not specified by this ECMA Standard. The scope of this `regid` includes the contents of this descriptor.

Note 21

The scope of the `regid` does not include the file; thus file-specific information recorded in this field may become out of date when the file is modified, particularly if multiple File Identifier Descriptors refer to the file.

14.4.8 File Identifier (RBP [L_IU+38])

This field shall specify an identification for the file described by the ICB identified in the ICB field.

14.4.9 Padding (RBP [L_FI+L_IU+38])

This field shall be $4 \times ip((L\_FI+L\_IU+38+3)/4) - (L\_FI+L\_IU+38)$ bytes long and shall contain all #00 bytes.

14.5 Allocation Extent Descriptor

The Allocation Extent Descriptor shall be recorded in the format shown in figure 4/14.

BP	Length	Name	Contents
0	16	Descriptor Tag	`tag` (4/7.2)(Tag=258)
16	4	Previous Allocation Extent Location	`Uint32` (1/7.1.5)
20	4	Length of Allocation Descriptors (=L_AD)	`Uint32` (1/7.1.5)

Figure 14 - Allocation Extent Descriptor format

14.5.1 Descriptor Tag (BP 0)

The Tag Identifier field of the `tag` (4/7.2) for this descriptor shall contain 258.

14.5.2 Previous Allocation Extent Location (BP 16)

This field shall specify the address, within the partition the descriptor is recorded on, of the previous allocation extent. If the extent's length is 0, the previous allocation extent is not specified.

14.5.3 Length of Allocation Descriptors (=L_AD) (BP 20)

This field specifies the length, in bytes, of the allocation descriptors recorded after this descriptor.

14.6 ICB Tag

All the entries in an ICB shall have a common format; a `tag` (4/7.2), followed by an `icbtag` as shown in figure 4/15, followed by a part that is unique to each type of entry and is described in the definition of that entry.

RBP	Length	Name	Contents
0	4	Prior Recorded Number of Direct Entries	`Uint32` (1/7.1.5)
4	2	Strategy Type	`Uint16` (1/7.1.3)
6	2	Strategy Parameter	bytes
8	2	Maximum Number of Entries	`Uint16` (1/7.1.3)
10	1	Reserved	#00 byte
11	1	File Type	`Uint8` (1/7.1.1)
12	6	Parent ICB Location	`lb_addr` (4/7.1)
18	2	Flags	`Uint16` (1/7.1.3)

Figure 15 - `icbtag` format

14.6.1 Prior Recorded Number of Direct Entries (RBP 0)

This field specifies the number of Direct Entries recorded in this ICB hierarchy prior to this entry.

14.6.2 Strategy Type (RBP 4)

This field shall specify the strategy for building the ICB hierarchy of which the ICB is a member. The strategies are specified by a number as shown in figure 4/16.

Type	Interpretation
0	The strategy is not specified by this clause.
1	The strategy is specified in 4/A.2.
2	The strategy is specified in 4/A.3.
3	The strategy is specified in 4/A.4.
4	The strategy is specified in 4/A.5.
5-4 095	Reserved for future standardisation.
4 096-65 535	The interpretation of the strategy shall be subject to agreement between the originator and recipient of the medium.

Figure 16 - ICB strategies

14.6.3 Strategy Parameter (RBP 6)

This field shall be interpreted according to the strategy specified by the Strategy Type field.

14.6.4 Maximum Number of Entries (RBP 8)

This field specifies the maximum number of entries, including both direct and indirect, that may be recorded in this ICB. This field shall be greater than 0.

14.6.5 Reserved (RBP 10)

This field shall be reserved for future standardisation and shall be set to 0.

14.6.6 File Type (RBP 11)

This field shall specify the type of the file as shown in figure 4/17.

Type	Interpretation
0	Shall mean that the interpretation of the file is not specified by this field
1	Shall mean that this is an Unallocated Space Entry (see 4/14.11)
2	Shall mean that this is a Partition Integrity Entry (see 4/14.13)
3	Shall mean that this is an Indirect Entry (see 4/14.7)
4	Shall mean that the file is a directory (see 4/8.6)
5	Shall mean that the file shall be interpreted as a sequence of bytes, each of which may be randomly accessed
6	Shall mean that the file is a block special device file as specified by ISO/IEC 9945-1
7	Shall mean that the file is a character special device file as specified by ISO/IEC 9945-1
8	Shall mean that the file is for recording Extended Attributes as described in 4/9
9	Shall mean that the file is a FIFO file as specified by ISO/IEC 9945-1
10	Shall mean that the file shall be interpreted according to the `C_ISSOCK` file type identified by ISO/IEC 9945-1
11	Shall mean that this is a Terminal Entry (see 4/14.8)
12	Shall mean that the file is a symbolic link and that its content is a pathname (see 4/8.7) for a file or directory
13-255	Reserved for future standardisation

Figure 17 - File Types

The interpretation of the content of files of File Types 0 and 5 shall be subject to agreement between the originator and recipient of the medium.

14.6.7 Parent ICB Location (RBP 12)

This field shall specify the location of the ICB which contains an indirect entry specifying the ICB that this descriptor is recorded in. If this field contains 0, no such ICB is specified.

14.6.8 Flags (RBP 18)

This field shall specify recording information about the file as shown in figure 4/8.

Bit	Interpretation
0-2	Shall be interpreted as a 3-bit unsigned binary number as follows. The value 0 shall mean that Short Allocation Descriptors (4/14.14.1) are used. The value 1 shall mean that Long Allocation Descriptors (4/14.14.2) are used. The value 2 shall mean that Extended Allocation Descriptors (4/14.14.3) are used. The value 3 shall mean that the file shall be treated as though it had exactly one allocation descriptor describing an extent which starts with the first byte of the Allocation Descriptors field and has a length, in bytes, recorded in the Length of Allocation Descriptors field. The values of 4-7 are reserved for future standardisation.
3	If the file is not a directory, this bit shall be reserved for future standardisation and set to ZERO. If the file is a directory and this bit is set to ONE, the directory shall be sorted according to 4/8.6.1. If the file is a directory and this bit is set to ZERO, the directory need not be sorted according to 4/8.6.1.
4	Non-relocatable: If set to ZERO, shall mean that there are no restrictions on how the allocation descriptors specifying the file's data may be modified; If set to ONE, the allocation descriptors shall not be modified such that either the address of an extent of the file is changed or that the recorded length of an extent is reduced.
5	Archive: This bit shall be set to ONE when the file is created or is written. This bit shall be set to ZERO in an implementation-dependent manner.
6	Setuid: This bit shall be interpreted as the `S_ISUID` bit as specified in ISO/IEC 9945-1.
7	Setgid: This bit shall be interpreted as the `S_ISGID` bit as specified in ISO/IEC 9945-1.
8	Sticky: This bit shall be interpreted as the `C_ISVTX` bit as specified in ISO/IEC 9945-1.
9	Contiguous: If set to ZERO, then an extent of a file need not begin at the first logical block after the last logical block of the preceding extent of the file; If set to ONE, then each extent of a file shall begin at the first logical block after the last logical block of the preceding extent of the file.
10	System: This bit shall be reserved for implementation use.
11	Transformed: If set to ZERO, shall mean that the recorded bytes of the data space of the file are those supplied by the user. If set to ONE, shall mean that the bytes supplied by the user have been transformed in a manner not specified by this ECMA Standard prior to recording.
12	Multi-versions: If the file is not a directory, this bit shall be reserved for future standardisation and shall be set to ZERO. If the file is a directory and the bit is set to ZERO, then no two File Identifier Descriptors recorded in the directory shall have the same contents of their File Identifier fields. If the file is a directory and the bit is set to ONE, then there may be two or more File Identifier Descriptors recorded in the directory with identical contents of their File Identifier fields.
13-15	Shall be reserved for future standardisation and all bits shall be set to ZERO.

Figure 18 - File characteristics

14.7 Indirect Entry

An Indirect Entry shall be recorded in the format shown in figure 4-19.

BP	Length	Name	Contents
0	16	Descriptor Tag	`tag` (4/7.2) (Tag=259)
16	20	ICB Tag	`icbtag` (4/14.6) (Type=3)
36	16	Indirect ICB	`long_ad` (4/14.14.2)

Figure 19 - Indirect Entry format

14.7.1 Descriptor Tag (BP 0)

The Tag Identifier field of the `tag` (4/7.2) for this descriptor shall contain 259.

14.7.2 ICB Tag (BP 16)

The File Type field of the `icbtag` (4/14.6) for this descriptor shall contain 3.

14.7.3 Indirect ICB (BP 36)

This field shall specify the address of another ICB

14.8 Terminal Entry

A Terminal Entry shall be recorded in the format shown in figure 4-20.

BP	Length	Name	Contents
0	16	Descriptor Tag	`tag` (4/7.2) (Tag=260)
16	20	ICB Tag	`icbtag` (4/14.6) (Type=11)

Figure 20 - Terminal Entry format

14.8.1 Descriptor Tag (BP 0)

The Tag Identifier field of the `tag` (4/7.2) for this descriptor shall contain 260.

14.8.2 ICB Tag (BP 16)

The File Type field of the `icbtag` (4/14.6) for this descriptor shall contain 11.

14.9 File Entry

The File Entry is a direct entry recorded in an ICB in the format shown in figure 4/ 21 for File Types 0 and 4-10 as specified in 4/14.6.6.

BP	Length	Name	Contents
0	16	Descriptor Tag	`tag` (4/7.2)(Tag=261)
16	20	ICB Tag	`icbtag` (4/14.6)
36	4	Uid	`Uint32` (1/7.1.5)
40	4	Gid	`Uint32` (1/7.1.5)
44	4	Permissions	`Uint32` (1/7.1.5)
48	2	File Link Count	`Uint16` (1/7.1.3)
50	1	Record Format	`Uint8` (1/7.1.1)
51	1	Record Display Attributes	`Uint8` (1/7.1.1)
52	4	Record Length	`Uint32` (1/7.1.5)
56	8	Information Length	`Uint64` (1/7.1.7)
64	8	Logical Blocks Recorded	`Uint64` (1/7.1.7)
72	12	Access Date and Time	`timestamp` (1/7.3)
84	12	Modification Date and Time	`timestamp` (1/7.3)
96	12	Attribute Date and Time	`timestamp` (1/7.3)
108	4	Checkpoint	`Uint32` (1/7.1.5)
112	16	Extended Attribute ICB	`long_ad` (4/14.14.2)
128	32	Implementation Identifier	`regid` (1/7.4)
160	8	Unique Id	`Uint64` (1/7.1.7)
168	4	Length of Extended Attributes (=L_EA)	`Uint32` (1/7.1.5)
172	4	Length of Allocation Descriptors (=L_AD)	`Uint32` (1/7.1.5)
176	L_EA	Extended Attributes	bytes
[L_EA+176]	L_AD	Allocation descriptors	bytes

Figure 21 - File Entry format

14.9.1 Descriptor Tag (BP 0)

The Tag Identifier field of the `tag` (4/7.2) for this descriptor shall contain 261.

14.9.2 ICB Tag (BP 16)

The File Type field of the `icbtag` (4/14.6) for this descriptor shall be recorded as specified in 4/14.6.6.

14.9.3 Uid (BP 36)

This field shall specify the user ID of the owner of the file.

Note 22

Originating systems that do not support the notion of user IDs will probably use an arbitrary user ID (and group ID). For various historical reasons, it is recommended such systems do not choose 0 for these IDs.

14.9.4 Gid (BP 40)

This field shall specify the group ID of the owner of the file.

14.9.5 Permissions (BP 44)

This field shall specify the access allowed to the current file for certain classes of users as follows:

- If the user's user ID is the same as the Uid field, then bits 10-14 shall apply.
- Otherwise, if the user's group ID is the same as the Gid field, then bits 5-9 shall apply.
- Otherwise, bits 0-4 shall apply.

The allowed access is shown in figure 422.

Bit	Interpretation
0	Other: If set to ZERO, shall mean that the user may not execute the file; If set to ONE, shall mean that the user may execute the file.
1	Other: If set to ZERO, shall mean that the user may not write the file; If set to ONE, shall mean that the user may write the file.
2	Other: If set to ZERO, shall mean that the user may not read the file; If set to ONE, shall mean that the user may read the file.
3	Other: If set to ZERO, shall mean that the user may not change any attributes of the file; If set to ONE, shall mean that the user may change attributes of the file.
4	Other: If set to ZERO, shall mean that the user may not delete the file; If set to ONE, shall mean that the user may delete the file.
5	Group: If set to ZERO, shall mean that the user may not execute the file; If set to ONE, shall mean that the user may execute the file.
6	Group: If set to ZERO, shall mean that the user may not write the file; If set to ONE, shall mean that the user may write the file.
7	Group: If set to ZERO, shall mean that the user may not read the file; If set to ONE, shall mean that the user may read the file.
8	Group: If set to ZERO, shall mean that the user may not change any attributes of the file; If set to ONE, shall mean that the user may change attributes of the file.
9	Group: If set to ZERO, shall mean that the user may not delete the file; If set to ONE, shall mean that the user may delete the file.
10	Owner: If set to ZERO, shall mean that the user may not execute the file; If set to ONE, shall mean that the user may execute the file.
11	Owner: If set to ZERO, shall mean that the user may not write the file; If set to ONE, shall mean that the user may write the file.
12	Owner: If set to ZERO, shall mean that the user may not read the file; If set to ONE, shall mean that the user may read the file.
13	Owner: If set to ZERO, shall mean that the user may not change any attributes of the file; If set to ONE, shall mean that the user may change attributes of the file.
14	Owner: If set to ZERO, shall mean that the user may not delete the file; If set to ONE, shall mean that the user may delete the file.
15-31	Reserved: Shall be set to ZERO.

Figure 22 - Allowed access

Note 23

File access schemes are subject to agreement between the originator and recipient of the medium as the meanings of both user IDs and group IDs are implementation dependent; indeed, the permission and file access models of the receiving and originating systems may be incompatible.

The question of how to interpret permissions on systems which do not support user IDs and group IDs is outside the scope of Part 4. However, if a system uses the Uid, Gid and Permissions fields, it is recommended that such systems use and set all three (owner, group, other) sets of permissions. It is also recommended that the Uid, Gid and Permissions fields be mapped to the appropriate fields in the implementation.

14.9.6 File Link Count (BP 48)

This field shall specify the number of File Identifier Descriptors identifying this ICB.

Note 24

Implementations should not blindly copy the contents of this field from the source File Entry when copying a directory hierarchy onto a volume. This field should only be incremented as links are made.

14.9.7 Record Format (BP 50)

This field shall specify a number identifying the format of the information in the file as shown in figure 23.

Number	Interpretation
0	Shall mean that the structure of the information recorded in the file is not specified by this field.
1	Shall mean that the information in the file is a sequence of padded fixed-length records (see 5/9.2.1).
2	Shall mean that the information in the file is a sequence of fixed-length records (see 5/9.2.2).
3	Shall mean that the information in the file is a sequence of variable-length-8 records (see 5/9.2.3.1).
4	Shall mean that the information in the file is a sequence of variable-length-16 records (see 5/9.2.3.2).
5	Shall mean that the information in the file is a sequence of variable-length-16-MSB records (see 5/9.2.3.3).
6	Shall mean that the information in the file is a sequence of variable-length-32 records (see 5/9.2.3.4).
7	Shall mean that the information in the file is a sequence of stream-print records (see 5/9.2.4).
8	Shall mean that the information in the file is a sequence of stream-LF records (see 5/9.2.5).
9	Shall mean that the information in the file is a sequence of stream-CR records (see 5/9.2.6).
10	Shall mean that the information in the file is a sequence of stream-CRLF records (see 5/9.2.7).
11	Shall mean that the information in the file is a sequence of stream-LFCR records (see 5/9.2.8).
12-255	Reserved for future standardisation.

Figure 23 - Information format

If the File Type field of the ICB Tag field contains 1, 2, 3, 4, 8, 11 or 12, the Record Format field shall contain 0.

14.9.8 Record Display Attributes (BP 51)

This field shall specify certain intended display attributes of the records in a file as shown in figure 24.

Attributes	Interpretation
0	Shall mean that the manner of display of a record is not specified by this field.
1	Shall mean that each record shall be displayed on a character-imaging device according to 5/9.3.1.
2	Shall mean that each record shall be displayed on a character-imaging device according to 5/9.3.2.
3	Shall mean that each record shall be displayed on a character-imaging device according to 5/9.3.3.
4-255	Reserved for future standardisation.

Figure 24 - Record display characteristics

14.9.9 Record Length (BP 52)

If the Record Format field contains the number 0, the interpretation of the Record Length field is subject to agreement between the originator and recipient of the medium.

If the Record Format field contains either 1 or 2, the Record Length field shall specify the length, in bytes, of each record in the file.

If the Record Format field contains a number in the range 3-11 inclusive, the Record Length field shall specify the maximum length, in bytes, of a record that may be recorded in the file.

14.9.10 Information Length (BP 56)

The file size in bytes. This shall be equal to the sum of the Information Lengths of the allocation descriptors for the body of the file (see 4/8.8.2 and 4/12).

Note 25

This is not necessarily the number of recorded bytes. There may be unrecorded extents or there may be `ext_ad` *(4/14.14.3) allocation descriptors.*

14.9.11 Logical Blocks Recorded (BP 64)

The number of recorded logical blocks specified by the allocation descriptors for the body of the file (see 4/12.1).

14.9.12 Access Date and Time (BP 72)

This field shall specify the most recent date and time of the day of file creation or read access to the file prior to recording this File Entry. This date and time shall not be earlier than the File Creation Date and Time specified in the File Times Extended Attribute, if any.

Note 26

This departs a little from the interpretation in ISO/IEC 9945-1 in that read accesses since this File Entry was recorded are ignored. This is intended to reduce updates on write-once media.

14.9.13 Modification Date and Time (BP 84)

This field shall specify the most recent date and time of the day of file creation or write access to the file. This date and time shall not be earlier than the File Creation Date and Time specified in the File Times Extended Attribute, if any.

14.9.14 Attribute Date and Time (BP 96)

This field shall specify the most recent date and time of the day of file creation or modification of the attributes of the file. This date and time shall not be earlier than the File Creation Date and Time specified in the File Times Extended Attribute, if any.

14.9.15 Checkpoint (BP 108)

This field shall contain 1 for the first instance of a file and shall be incremented by 1 when directed to do so by the user. Part 4 does not specify any relationship between the Checkpoint field and the File Version Number field of the directory descriptor identifying the file.

Note 27

This field allows the user to label sequences of instances of a file with a monotonic increasing numeric tag. It has an interpretation similar to that of the File Version Number but is not part of the file's identification and need not have the same value as the File Version Number. The motivation is that the user will often have no control over when an implementation will flush a file to disk (and thus creating a new instance). In this situation, the user may simply increment the Checkpoint field when it is appropriate.

14.9.16 Extended Attribute ICB (BP 112)

This field shall specify the ICB describing the extended attribute file for the file. If the extent's length is 0, no such ICB is specified.

14.9.17 Implementation Identifier (BP 128)

This field shall specify an identification of an implementation which can recognise and act upon the contents of the Implementation Use field, if any, of the allocation descriptors for this File Entry. If this field contains all #00 bytes, then no such implementation is identified. The scope of this `regid` includes the contents of the descriptors that specify the contents and attributes of the file described by this descriptor.

14.9.18 Unique Id (BP 160)

This field shall specify a numeric identifier for this file. All File Entries with the same contents of this field shall describe the same file or directory

14.9.19 Length of Extended Attributes (=L_EA) (BP 168)

This field shall specify the length, in bytes, of the Extended Attributes field. L_EA shall be an integral multiple of 4.

14.9.20 Length of Allocation Descriptors (=L_AD) (BP 172)

This field shall specify the length, in bytes, of the Allocation Descriptors field.

14.9.21 Extended Attributes (BP 176)

This field shall contain an extended attributes space (see 4/9). The recorded extended attributes shall occupy at most L_EA bytes and any unused bytes shall be set to #00.

14.9.22 Allocation Descriptors (BP [L_EA+176])

This field shall be a sequence of allocation descriptors recorded as specified in 4/12.1. Any such allocation descriptor which is specified as unrecorded and unallocated (see 4/14.14.1.1) shall have its Extent Location field set to 0.

14.10 Extended Attributes

In this clause, the term "current file" shall refer to the file that the extended attribute is associated with.

14.10.1 Extended Attribute Header Descriptor

The Extended Attribute Header Descriptor shall be recorded in the format shown in figure 4-25.

RBP	Length	Name	Contents
0	16	Descriptor Tag	`tag` (4/7.2)(Tag=262)
16	4	Implementation Attributes Location	`Uint32` (1/7.1.5)
20	4	Application Attributes Location	`Uint32` (1/7.1.5)

Figure 25 - Extended Attribute Header Descriptor format

14.10.1.1 Descriptor Tag (RBP 0)

The Tag Identifier field of the `tag` (4/7.2) for this descriptor shall contain 262.

14.10.1.2 Implementation Attributes Location (RBP 16)

This field shall specify the start of the implementation use extended attributes as a byte offset from the start of an extended attributes space in which extended attributes of the current file shall be recorded.

14.10.1.3 Application Attributes Location (RBP 20)

This field shall specify the start of the application use extended attributes as a byte offset from the start of an extended attributes space in which extended attributes of the current file shall be recorded.

14.10.2 Generic format

An Extended Attribute shall be recorded in the format shown in figure 4/26. The specification for each extended attribute shall specify the interpretation of the Attribute Subtype and Attribute Data fields of the extended attribute.

RBP	Length	Name	Contents
0	4	Attribute Type	`Uint32` (1/7.1.5)
4	1	Attribute Subtype	`Uint8` (1/7.1.1)
5	3	Reserved	#00 bytes
8	4	Attribute Length (=A_L)	`Uint32` (1/7.1.5)
12	A_L-12	Attribute Data	bytes

Figure 26 - Generic extended attributeformat

14.10.2.1 Attribute Type (RBP 0)

This field shall specify the type of the extended attribute

14.10.2.2 Attribute Subtype (RBP 4)

This field shall specify the subtype of the extended attribute

14.10.2.3 Reserved (RBP 5)

This field shall be reserved for future standardisation and all bytes shall be set to #00.

14.10.2.4 Attribute Length (=A_L) (RBP 8)

This field shall specify the length of the entire extended attribute

Note 28

It is recommended that the extended attributelength be an integral multiple of 4.

14.10.2.5 Attribute Data (RBP 12)

The interpretation of this field shall depend on the value of the Attribute Typefield.

Note 29

The only meaning for the Attribute Length field (A_L) is the distance in bytes from the start of the extended attribute to the start of the next, if any, extended attribute. The only deduction one can make is that the amount of attribute specific data is not greater than A_L-12. It is recommended that extended attributes with variable-sized data record the data length immediately after the Attribute Length field.

This scheme allows for arbitrary alignment of the attributes and their data. In particular, there may be padding bytes between the end of the data for an attribute and the start of the next attribute. Implementations are not required to preserve any attribute alignments.

14.10.3 Character Set Information

The Character Set Information Extended Attribute shall be recorded in the format shown in figure 4/27. The Character Set Information Extended Attribute may be used to specify the coded character sets used in interpreting the contents of the current file

RBP	Length	Name	Contents
0	4	Attribute Type	`Uint32` (1/7.1.5) = 1
4	1	Attribute Subtype	`Uint8` (1/7.1.1) = 1
5	3	Reserved	#00 bytes
8	4	Attribute Length	`Uint32` (1/7.1.5)
12	4	Escape Sequences Length (=ES_L)	`Uint32` (1/7.1.5)
16	1	Character Set Type	`Uint8` (1/7.1.1)
17	ES_L	Escape Sequences	bytes

Figure 27 - Character Set Information Extended Attribute format

14.10.3.1 Attribute Type (RBP 0)

This field shall specify 1.

14.10.3.2 Attribute Subtype (RBP 4)

This field shall specify 1. All other values are reserved for future standardisation.

14.10.3.3 Reserved (RBP 5)

This field shall be reserved for future standardisation and all bytes shall be set to #00.

14.10.3.4 Attribute Length (RBP 8)

This field shall specify the length of the entire extended attribute

Note 30

It is recommended that the extended attribute length be an integral multiple of 4.

14.10.3.5 Escape Sequences Length (=ES_L) (RBP 12)

This field shall specify the length in bytes of the Escape Sequences field.

14.10.3.6 Character Set Type (RBP 16)

This field shall specify the character set type as specified in 1/7.2.1, except that any information that would be recorded in the Character Set Information field shall instead be recorded in the Escape Sequences field.

14.10.3.7 Escape Sequences (RBP 17)

This field shall specify one or more escape sequences, control sequences or both escape sequences and control sequences according to ECMA-35 and ECMA-48 that designate and implicitly invoke the coded character sets to be used in interpreting the conents of the current file in an 8-bit environment according to ECMA-35 or ISO/IEC 10646-1. These sequences shall be recorded contiguously from the start of the field and any unused bytes shall be set to #00.

14.10.4 Alternate Permissions

The Alternate Permissions extended attribute specifies fields that can be used to support the file access permission scheme of ECMA-119 for the current file. It shall be recorded in the format shown in figure 428.

Note 31

This extended attribute is an extension of the permissions field in ECMA-119 to allow for the case of writing information. In addition, it eliminates the inconsistencies of the specification in ECMA-119.

If the user's user ID is the same as the Owner Identification field and the user's group ID is the same as the Group Identification field, the user shall be treated as the owner of the file.

RBP	Length	Name	Contents
0	4	Attribute Type	`Uint32` (1/7.1.5) = 3
4	1	Attribute Subtype	`Uint8` (1/7.1.1) = 1
5	3	Reserved	#00 bytes
8	4	Attribute Length	`Uint32` (1/7.1.5)
12	2	Owner Identification	`Uint16` (1/7.1.3)
14	2	Group Identification	`Uint16` (1/7.1.3)
16	2	Permission	`Uint16` (1/7.1.3)

Figure 28 - Alternate Permissions Extended Attribute format

14.10.4.1 Attribute Type (RBP 0)

This field shall specify 3.

14.10.4.2 Attribute Subtype (RBP 4)

This field shall specify 1. All other values are reserved for future standardisation.

14.10.4.3 Reserved (RBP 5)

This field shall be reserved for future standardisation and all bytes shall be set to #00.

14.10.4.4 Attribute Length (RBP 8)

This field shall specify the length of the entire extended attribute

Note 32

It is recommended that the extended attribute length be an integral multiple of 4.

14.10.4.5 Owner Identification (RBP 12)

This field shall specify as a 16-bit number an identification of the owner of the file who is a member of the group identified by the Group Identification field of this extended attribute

If the number in this field is 0, this shall indicate that there is no owner identification specified for the file. In this case, the Group Identification field shall be set to 0.

14.10.4.6 Group Identification (RBP 14)

This field shall specify as a 16-bit number an identification of the group of which the owner of the file is a member.

For this number, values from 1 to a value subject to agreement between the originator and recipient of the medium shall identify the group as belonging to the class of user referred to as System.

If the number in this field is 0, this shall indicate that there is no group id entification specified for the file. In this case, the Owner Identification field shall be set to 0.

14.10.4.7 Permissions (RBP 16)

This field shall specify, for certain classes of users, if read, write, execute, and delete access is allowed for the file. The desired access shall be given if at least one of the following conditions is true:

- the user's user ID is the same as the Owner Identification field and the user's group ID is the same as the Group Identification field and bits 4-7 allow the desired access,
- bits 12-15 allow the desired access,
- the user's group ID is the same as the Group Identification field and bits 8-11 allow the desired access,
- if the user's group ID identifies a group of the System class of user and bits 0-3 allow the desired access.

The allowed access is shown in figure 429.

Bit	Interpretation
0	If set to ZERO, shall mean that a user who is a member of a group of the System class of user may read the file. If set to ONE, shall mean that read access is not allowed by this bit.
1	If set to ZERO, shall mean that a user who is a member of a group of the System class of user may write the file. If set to ONE, shall mean that write access is not allowed by this bit.
2	If set to ZERO, shall mean that a user who is a member of a group of the System class of user may execute the file. If set to ONE, shall mean that execute access is not allowed by this bit.
3	If set to ZERO, shall mean that a user who is a member of a group of the System class of user may delete the file. If set to ONE, shall mean that delete access is not allowed by this bit.
4	If set to ZERO, shall mean that the owner may read the file. If set to ONE, shall mean that read access is not allowed by this bit.
5	If set to ZERO, shall mean that the owner may write the file. If set to ONE, shall mean that write access is not allowed by this bit.
6	If set to ZERO, shall mean that the owner may execute the file. If set to ONE, shall mean that execute access is not allowed by this bit.
7	If set to ZERO, shall mean that the owner may delete the file. If set to ONE, shall mean that delete access is not allowed by this bit.
8	If set to ZERO, shall mean that any user who has a group ID that is the same as the Group Identification field may read the file. If set to ONE, shall mean that read access is not allowed by this bit.
9	If set to ZERO, shall mean that any user who has a group ID that is the same as the Group Identification field may write the file. If set to ONE, shall mean that write access is not allowed by this bit.
10	If set to ZERO, shall mean that any user who has a group ID that is the same as the Group Identification field may execute the file. If set to ONE, shall mean that execute access is not allowed by this bit.
11	If set to ZERO, shall mean that any user who has a group ID that is the same as the Group Identification field may delete the file. If set to ONE, shall mean that delete access is not allowed by this bit.
12	If set to ZERO, shall mean that any user may read the file. If set to ONE, shall mean that read access is not allowed by this bit.
13	If set to ZERO, shall mean that any user may write the file. If set to ONE, shall mean that write access is not allowed by this bit.
14	If set to ZERO, shall mean that any user may execute the file. If set to ONE, shall mean that execute access is not allowed by this bit.
15	If set to ZERO, shall mean that any user may delete the file. If set to ONE, shall mean that delete access is not allowed by this bit.

Figure 29 - Allowed access

Note 33

File access schemes are subject to agreement between the originator and recipient of the medium as the meanings of both user IDs and group IDs are implementation dependent; indeed, the permission and file access models of the receiving and originating systems may be incompatible.

The question of how to interpret permissions on systems which do not support user IDs and group IDs is outside the scope of Part 4. However, if a system uses the Alternate Permissions extended attribute, it is recommended that such systems use and set all four (system, owner, group, other) sets of permissions. It is also recommended that the Owner Identification, Group Identification and Permission fields be mapped to the appropriate fields in the implementation.

14.10.5 File Times Extended Attribute

The File Times Extended Attribute specifies certain dates and times for the current file and shall be recorded as shown in figure 4/30.

RBP	Length	Name	Contents
0	4	Attribute Type	`Uint32` (1/7.1.5) = 5
4	1	Attribute Subtype	`Uint8` (1/7.1.1) = 1
5	3	Reserved	#00 bytes
8	4	Attribute Length	`Uint32` (1/7.1.5
12	4	Data Length(=D_L)	`Uint32` (1/7.1.5)
16	4	File Time Existence	`Uint32` (1/7.1.5)
20	D_L	File Times	bytes

Figure 30 - File Times Extended Attributeformat

14.10.5.1 Attribute Type (RBP 0)

This field shall specify 5.

14.10.5.2 Attribute Subtype (RBP 4)

This field shall specify 1. All other values are reserved for future standardisation.

14.10.5.3 Reserved (RBP 5)

This field shall be reserved for future standardisation and all bytes shall be set to #00.

14.10.5.4 Attribute Length (RBP 8)

This field shall specify the length of the entire extended attribute

Note 34

It is recommended that the extended attribute length be an integral multiple of 4.

14.10.5.5 Data Length(=D_L) (RBP 12)

This field shall contain the number of bytes used to record the dates and times specified by the File Time Existence field.

14.10.5.6 File Time Existence (RBP 16)

This field shall specify which dates and times shall be recorded in the File Times field. A bit in this field corresponds to a particular date and time as shown in figure 4/ 31. If the bit is ZERO, then that date and time shall not be recorded. If the bit is ONE, then that date and time shall be recorded. Bits not specified in figure 4/31 are reserved for future standardisation and shall be set to ZERO.

Bit	Interpretation
0	File Creation Date and Time: the date and time of the day at which the file was created.
2	File Deletion Date and Time: the date and time of the day after which the file may be deleted. If the bit is ZERO, the file may not be deleted unless requested by the user.
3	File Effective Date and Time: the date and time of the day after which the file may be used. If the bit is ZERO, the file may be used at once.
5	File Last Backup Date and Time: the date and time of the day at which the file was last backed up.

Figure 31 - File Times

Note 35

Bits 1 and 4 are deliberately unused for compatibility with ECMA-168. Attribute type 5 in ECMA-168 also specifies File Last Access Date and Time and File Modification Date and Time. Those dates and times are specified in the File Entry (see 4/14.9.12 and 4/14.9.13) of Part 4.

14.10.5.7 File Times (RBP 20)

The dates and times specified in the File Times Existence field shall be recorded contiguously in this field, each as a `timestamp` (1/7.3), in ascending order of their bit positions.

14.10.6 Information Times Extended Attribute

The Information Times Extended Attribute specifies certain dates and times for the information in the current file and shall be recorded as shown in figure 432.

RBP	Length	Name	Contents
0	4	Attribute Type	`Uint32` (1/7.1.5) = 6
4	1	Attribute Subtype	`Uint8` (1/7.1.1) = 1
5	3	Reserved	#00 bytes
8	4	Attribute Length	`Uint32` (1/7.1.5)
12	4	Data Length(=D_L)	`Uint32` (1/7.1.5)
16	4	Information Time Existence	`Uint32` (1/7.1.5)
20	D_L	Information Times	bytes

Figure 32 - Information Times Extended Attribute format

14.10.6.1 Attribute Type (RBP 0)

This field shall specify 6.

14.10.6.2 Attribute Subtype (RBP 4)

This field shall specify 1. All other values are reserved for future standardisation.

14.10.6.3 Reserved (RBP 5)

This field shall be reserved for future standardisation and all bytes shall be set to #00.

14.10.6.4 Attribute Length (RBP 8)

This field shall specify the length of the entire extended attribute.

Note 36

It is recommended that the extended attribute length be an integral multiple of 4.

14.10.6.5 Data Length(=D_L) (RBP 12)

This field shall contain the number of bytes used to record the dates and times specified by the Information Time Existence field.

14.10.6.6 Information Time Existence (RBP 16)

This field shall specify which dates and times shall be recorded in the Information Times field. A bit in this field corresponds to a particular date and time as shown in figure 4/ 33. If the bit is ZERO, then that date and time shall not be recorded. If the bit is ONE, then that date and time shall be recorded. Bits not specified in figure 4/33 are reserved for future standardisation and shall be set to ZERO.

Bit	Interpretation
0	Information Creation Date and Time: the date and time of the day at which the information in the file was created.
1	Information Last Modification Date and Time: the date and time of the day at which the information in the file was last modified.
2	Information Expiration Date and Time: the date and time of the day after which the information in the file may be regarded as obsolete. If the bit is ZERO, the information in the file shall not be regarded as obsolete unless requested by the user.
3	Information Effective Date and Time: the date and time of the day after which the information in the file may be used. If the bit is ZERO, the information in the file may be used at once.

Figure 33 - Information Times

14.10.6.7 Information Times (RBP 20)

The dates and times specified in the Information Times Existence field shall be recorded contiguously in this field, each as a `timestamp` (1/7.3), in ascending order of their bit positions

14.10.7 Device Specification

The Device Specification Extended Attribute shall be recorded in the format shown in figure 4/ 34. It shall specify a device subject to agreement between the originator and recipient of the medium.

RBP	Length	Name	Contents
0	4	Attribute Type	`Uint32` (1/7.1.5) = 12
4	1	Attribute Subtype	`Uint8` (1/7.1.1) = 1
5	3	Reserved	#00 bytes
8	4	Attribute Length	`Uint32` 1/7.1.5)
12	4	Implementation Use Length (=IU_L)	`Uint32` (1/7.1.5)
16	4	Major Device Identification	`Uint32` (1/7.1.5)
20	4	Minor Device Identification	`Uint32` (1/7.1.5)
24	IU_L	Implementation Use	bytes

Figure 34 - Device Specification Extended Attribute format

14.10.7.1 Attribute Type (RBP 0)

This field shall specify 12.

14.10.7.2 Attribute Subtype (RBP 4)

This field shall specify 1. All other values are reserved for future standardisation.

14.10.7.3 Reserved (RBP 5)

This field shall be reserved for future standardisation and all bytes shall be set to #00.

14.10.7.4 Attribute Length (RBP 8)

This field shall specify the length of the entire extended attribute.

Note 37

It is recommended that the extended attribute length be an integral multiple of 4.

14.10.7.5 Implementation Use Length (=IU_L) (RBP 12)

This field shall specify the length in bytes of the Implementation Use field.

14.10.7.6 Major Device Identification (RBP 16)

This field may be used to specify a device. The contents of this field shall be subject to agreement between the originator and recipient of the medium.

14.10.7.7 Minor Device Identification (RBP 20)

This field may be used to specify a device. The contents of this field shall be subject to agreement between the originator and recipient of the medium.

14.10.7.8 Implementation Use (RBP 24)

If IU_L is greater than 0, this field shall specify an identification of an implementation, recorded as a `regid` (1/7.4) in the first 32 bytes of this field, which can recognise and act upon the remainder of this field, which shall be reserved for implementation use and its contents are not specified by this ECMA Standard.

14.10.8 Implementation Use Extended Attribute

The Implementation Use Extended Attribute shall be recorded in the format shown in figure 35.

RBP	Length	Name	Contents
0	4	Attribute Type	`Uint32` (1/7.1.5) = 2 048
4	1	Attribute Subtype	`Uint8` (1/7.1.1) = 1
5	3	Reserved	#00 bytes
8	4	Attribute Length	`Uint32` (1/7.1.5)
12	4	Implementation Use Length (=IU_L)	`Uint32` (1/7.1.5)
16	32	Implementation Identifier	`regid` (1/7.4)
48	IU_L	Implementation Use	bytes

Figure 35 - Implementation Use Extended Attribute format

14.10.8.1 Attribute Type (RBP 0)

This field shall specify 2048.

14.10.8.2 Attribute Subtype (RBP 4)

This field shall specify 1. All other values are reserved for future standardisation.

14.10.8.3 Reserved (RBP 5)

This field shall be reserved for future standardisation and all bytes shall be set to #00.

14.10.8.4 Attribute Length (RBP 8)

This field shall specify the length of the entire extended attribute

Note 38

It is recommended that the extended attribute length be an integral multiple of 4.

14.10.8.5 Implementation Use Length (=IU_L) (RBP 12)

This field shall specify the length of the Implementation Use field.

14.10.8.6 Implementation Identifier (RBP 16)

This field shall specify an identification of an implementation which can recognise and act upon the contents of the Implementation Use field. If this field contains all #00 bytes, then no such implementation is identified. The scope of this `regid` includes the contents of the descriptors that specify the contents and attributes of the current file.

14.10.8.7 Implementation Use (RBP 48)

This field shall be reserved for implementation use. The interpretation of the contents of this field is not specified by Part 4.

14.10.9 Application Use Extended Attribute

The Application Use Extended Attribute shall be recorded in the format shown in figure 36.

RBP	Length	Name	Contents
0	4	Attribute Type	`Uint32` (1/7.1.5) = 65 536
4	1	Attribute Subtype	`Uint8` (1/7.1.1) = 1
5	3	Reserved	#00 bytes
8	4	Attribute Length	`Uint32` (1/7.1.5)
12	4	Application Use Length(=AU_L)	`Uint32` (1/7.1.5)
16	32	Application Identifier	`regid` (1/7.4)
48	AU_L	Application Use	bytes

Figure 36 - Application Use Extended Attribute format

14.10.9.1 Attribute Type (RBP 0)

This field shall specify 65536.

14.10.9.2 Attribute Subtype (RBP 4)

This field shall specify 1. All other values are reserved for future standardisation.

14.10.9.3 Reserved (RBP 5)

This field shall be reserved for future standardisation and all bytes shall be set to #00.

14.10.9.4 Attribute Length (RBP 8)

This field shall specify the length of the entire extended attribute

Note 39

It is recommended that the extended attribute length be an integral multiple of 4.

14.10.9.5 Application Use Length(=AU_L) (RBP 12)

This field shall specify the length of the Application Use field.

14.10.9.6 Application Identifier (RBP 16)

This field shall specify an identification of an application which can recognise and act upon the contents of the Application Use field. If this field contains all #00 bytes, then no such application is identified. The scope of this `regid` includes the contents of the descriptors that specify the contents and attributes of the current file.

14.10.9.7 Application Use (RBP 48)

This field shall be reserved for application use. The interpretation of the contents of this field is not specified by Part 4.

14.11 Unallocated Space Entry

An Unallocated Space Entry is a direct entry recorded within an ICB and shall be recorded in the format shown in figure 4/37.

Note 40

This is normally only used for write-once media.

BP	Length	Name	Contents
0	16	Descriptor Tag	`tag` (4/7.2) (Tag=263)
16	20	ICB Tag	`icbtag` (4/14.6) (Type=1)
36	4	Length of Allocation Descriptors (=L_AD)	`Uint32` (1/7.1.5)
40	L_AD	Allocation descriptors	bytes

Figure 37 - Unallocated Space Entry format

14.11.1 Descriptor Tag (BP 0)

The Tag Identifier field of the `tag` (4/7.2) for this descriptor shall contain 263.

14.11.2 ICB Tag (BP 16)

The File Type field of the `icbtag` (4/14.6) for this descriptor shall contain 1.

14.11.3 Length of Allocation Descriptors (=L_AD) (BP 36)

This field specifies the length, in bytes, of the Allocation Descriptor s field. L_AD+40 shall not be greater than the size of a logical block

14.11.4 Allocation Descriptors (BP 40)

This field shall contain allocation descriptors.

The type of allocation descriptor shall be specified by the Flags field in the ICB Tag field (see 4/14.6.8). The extent length fields of these allocation descriptors shall be an integral multiple of the logical block size.

14.12 Space Bitmap Descriptor

A Space Bitmap descriptor specifies a bit for every logical block in the partition and shall be recorded in the format shown in figure 4/38.

BP	Length	Name	Contents
0	16	Descriptor Tag	`tag` (4/7.2)(Tag=264)
16	4	Number of Bits (=N_BT)	`Uint32` (1/7.1.5)
20	4	Number of Bytes (=N_B)	`Uint32` (1/7.1.5)
24	N_B	Bitmap	bytes

Figure 38 - Space Bitmap Descriptor format

14.12.1 Descriptor Tag (BP 0)

The Tag Identifier field of the `tag` (4/7.2) for this descriptor shall contain 264.

14.12.2 Number of Bits (=N_BT) (BP 16)

This field shall specify the number of valid bits in the Bitmap field.

14.12.3 Number of Bytes (=N_B) (BP 20)

This field shall specify the number of bytes in the Bitmap field. The length of this field shall not be less than *ip((N_BT+7)/8)* bytes.

14.12.4 Bitmap (BP 24)

This field specifies a bit for each logical block in the partition. The bit for logical block *s* is bit *rem(s,8)* in byte *ip(s/8)*, where byte 0 is the first byte in this field.

14.13 Partition Integrity Entry

A Partition Integrity Entry is a direct entry recorded in an ICB and shall be recorded in the format shown in figure 4/39.

BP	Length	Name	Contents
0	16	Descriptor Tag	`tag` (4/7.2) (Tag=265)
16	20	ICB Tag	`icbtag` (4/14.6) (Type=2)
36	12	Recording Date and Time	`timestamp` (1/7.3)
48	1	Integrity Type	`Uint8` (1/7.1.1)
49	175	Reserved	#00 bytes
224	32	Implementation Identifier	`regid` (1/7.4)
256	256	Implementation Use	bytes

Figure 39 - Partition Integrity Entryformat

14.13.1 Descriptor Tag (BP 0)

The Tag Identifier field of the `tag` 4/7.2) for this descriptor shall contain 265.

14.13.2 ICB Tag (BP 16)

The File Type field of the `icbtag` (4/14.6) for this descriptor shall contain 2.

14.13.3 Recording Date and Time (BP 36)

This field shall specify the date and time of the day of recording of this Integrity Entry.

14.13.4 Integrity Type (BP 48)

This field shall specify the type of Integrity Entry. The types are shown in figure 40.

Type	Interpretation
0	Shall mean that the entry is an Open Integrity Entry.
1	Shall mean that the entry is a Close Integrity Entry.
2	Shall mean that the entry is a Stable Integrity Entry.
3-255	Reserved for future standardisation.

Figure 40 - Integrity Entry interpretation

14.13.5 Reserved (BP 49)

This field shall be reserved for future standardisation and all bytes shall be set to #00.

14.13.6 Implementation Identifier (BP 224)

This field shall specify an identification of an implementation which can recognise and act upon the contents of the Implementation Use field. If this field contains all #00 bytes, then no such implementation is identified. The scope of this `regid` includes the contents of the partition associated with this descriptor.

14.13.7 Implementation Use (BP 256)

This field shall be reserved for implementation use. Its content is not specified by this Part.

14.14 Allocation descriptors

14.14.1 Short Allocation Descriptor

The Short Allocation Descriptor, designated as `short_ad`, shall be recorded in the format shown in figure 4/41.

RBP	Length	Name	Contents
0	4	Extent Length	`Uint32` (1/7.1.5)
4	4	Extent Position	`Uint32` (1/7.1.5)

Figure 41 - `short_ad` format

14.14.1.1 Extent Length (RBP 0)

The 30 least significant bits of this field shall be interpreted as a 30-bit unsigned binary number specifying the length of the extent in bytes. Unless otherwise specified, the length shall be an integral multiple of the logical block size. The 2 most significant bits shall be interpreted as a 2-bit unsigned binary number specifying the type of the extent as described in figure 4/42.

Value	Interpretation
0	Extent recorded and allocated
1	Extent not recorded but allocated
2	Extent not recorded and not allocated
3	The extent is the next extent of allocation descriptors (see 4/12)

Figure 42 - Extent interpretation

14.14.1.2 Extent Position (RBP 4)

This field shall specify the logical block number, within the partition the descriptor is recorded on, of the extent. If the extent's length is 0, no extent is specified and this field shall contain 0.

14.14.2 Long Allocation Descriptor

The Long Allocation Descriptor, designated by `long_ad`, shall be recorded in the format shown in figure 4/43.

RBP	Length	Name	Contents
0	4	Extent Length	`Uint32` (1/7.1.5)
4	6	Extent Location	`lb_addr` (4/7.1)
10	6	Implementation Use	bytes

Figure 43 - `long_ad` format

14.14.2.1 Extent Length (RBP 0)

This field shall be recorded as specified in 4/14.14.1.1.

14.14.2.2 Extent Location (RBP 4)

This field shall specify the logical block number of the extent. If the extent's length is 0, no extent is specified and this field shall contain 0.

14.14.2.3 Implementation Use (RBP 10)

This field shall be reserved for implementation use. Its content is not specified by this Part.

Note 41

The `long_ad` (4/14.14.2) is intended for use when the extent's location may be on another partition (either on this volume or another).

14.14.3 Extended Allocation Descriptor

The Extended Allocation Descriptor, designated by `ext_ad`, shall be recorded in the format shown in figure 4/44.

RBP	Length	Name	Contents
0	4	Extent Length	`Uint32` (1/7.1.5)
4	4	Recorded Length	`Uint32` (1/7.1.5)
8	4	Information Length	`Uint32` (1/7.1.5)
12	6	Extent Location	`lb_addr` (4/7.1)
18	2	Implementation Use	bytes

Figure 44 - `ext_ad` format

14.14.3.1 Extent Length (RBP 0)

This field shall be recorded as specified in 4/14.14.1.1.

14.14.3.2 Recorded Length (RBP 4)

The two most significant bits of this field are reserved for future standardisation and shall be set to ZERO. The 30 least significant bits of this field shall be interpreted as a 30-bit unsigned binary number specifying the number of bytes recorded in the extent. This may be different from the number of bytes specified in the Extent Length field.

14.14.3.3 Information Length (RBP 8)

This field shall specify how many bytes of information are recorded starting at the first byte of the extent identified by the Extent Location field. This may be different from the value in either the Extent Length field or the Recorded Length field.

14.14.3.4 Extent Location (RBP 12)

This field shall specify the logical block number of the extent. If the extent's length is 0, no extent is specified and this field shall contain all #00 bytes.

14.14.3.5 Implementation Use (RBP 18)

This field shall be reserved for implementation use Its content is not specified by Part.

Note 42

The `ext_ad` (4/14.14.3) is similar to the `long_ad` (4/14.14.2) except that while Information Length bytes are represented in the extent, only Recorded Length bytes have been recorded. (Most likely, a compression algorithm has been applied on the extent.) The Recorded Length allows implementations to copy files (and their extents) without knowing how or why the Recorded Length differs from the Information Length.

14.15 Logical Volume Header Descriptor

The Logical Volume Header Descriptor shall specify a numeric file and directory identifier and shall be recorded with the format shown in figure 4/45 (see 4/3.1 for where this descriptor is recorded).

RBP	Length	Name	Contents
0	8	Unique Id	`Uint64` (1/7.1.7)
8	24	Reserved	#00 bytes

Figure 45 - Logical Volume Header Descriptor format

14.15.1 Unique Id (RBP 0)

This field shall specify a value which is greater than the value of the Unique Id field in any File Entry recorded on the associated logical volume.

Note 33

The intended use of this field is to facilitate allocation of unique identifiers for files and directories (see 4/14.9.18). In order to avoid having to examine every file and directory, this field should be maintained even if the rest of the volume is not conformant with this ECMA Standard. An implementation might record several Open Integrity Descriptors consecutively just to maintain this field as a modest increment over the last value recorded. Implementations should not assume any ordering properties for the value of this field; the value might decrease or increase with successive descriptors. In particular, the value in a Close Integrity Descriptor might be less than the value in the preceding Open Integrity Descriptor.

14.15.2 Reserved (RBP 8)

This field shall be reserved for future standardisation and all bytes shall contain #00.

14.16 Pathname

14.16.1 Path Component

A Path Component shall be recorded in the format shown in figure 46.

RBP	Length	Name	Contents
0	1	Component Type	`Uint8` (1/7.1.1)
1	1	Length of Component Identifier (= L_CI)	`Uint8` (1/7.1.1)
2	2	Component File Version Number	`Uint16` (1/7.1.3)
4	L_CI	Component Identifier	d-characters (1/7.2)

Figure 46 - Path Component format

14.16.1.1 Component Type (RBP 0)

This field shall specify the component type as shown in figure 47.

Type	Interpretation
0	Reserved for future standardisation.
1	If L_CI is not 0, the component specifies the root of a directory hierarchy subject to agreement between the originator and recipient of the medium. If L_CI is 0, this component shall specify the root of a file system as specified in ISO/IEC 9945-1.
2	The component specifies the root directory of the directory hierarchy of which the predecessor of the first component in the pathname is a member.
3	The component specifies the parent directory of the predecessor component.
4	The component specifies the same directory as the predecessor component.
5	The component identifies an object, either a file or a directory or an alias, specified by a descriptor of the directory identified by the predecessor component, such that the contents of the File Identifier field of that directory descriptor is identical to the contents of the Component Identifier field.
6-255	Reserved for future standardisation.

Figure 47 - Component interpretation

14.16.1.2 Length of Component Identifier (= L_CI) (RBP 1)

If the Component Type field contains 1 or 5, this field shall specify the length in bytes of the Component Identifier field. If the Component Type field contains 5, L_CI shall be greater than 0. If the Component Type field does not contain 1 or 5, this field shall contain 0.

14.16.1.3 Component File Version Number (RBP 2)

This field shall specify the file version number of the component as follows.

If the number in this field is 0, then the highest file version number of any instance of the entity identified by the Component Identifier field (see 4/8.7) is identified.

If the number in this field is in the range 1 to 32 767 inclusive, this field shall specify the file version number of the entity identified by the Component Identifier field (see 4/8.7). The numbers 32 768 to 65 535 inclusive are reserved for future standardisation.

If the entity identified by the Component Identifier field (see 4/8.7) is a directory , then the value of this field shall be 0.

Note 44

This allows versions of files and aliases to be specified in recorded pathnames.

14.16.1.4 Component Identifier (RBP 4)

This field shall identify the component.

15 Levels of medium interchange

Part 4 specifies three levels of medium interchange . The level of a file set shall be that level specifying the most restrictions required to record the file set according to the specifications of Part 4.

15.1 Level 1

At level 1, the following restrictionsshall apply:

- The number in any Length of File Identifier field shall not exceed 12.
- A File Identifier (see 4/14.4) for a directory shall conform to the schema shown in figure 4/48. A sequence of fileid-characters shall be a sequence of d-characters (1/7.2) excluding SPACE, COMMA, FULL STOP and REVERSE SOLIDUScharacters except as part of a code extension character(see 1/7.2.9.1).

```
[Directory File Identifier]{
       <fileid-characters>1+8
}
```

Figure 48 - Directory file identifier schema

- A File Identifier (see 4/14.4) for a non-directoryfile shall conform to the schemashown in figure 4/49.

```
[Nondirectory File Identifier]{
       <fileid-characters>1+8
   | {
            <fileid-characters>1+8
            <FULL STOP character>
            <fileid-characters>0+3
     }
   | {
            <fileid-characters>0+8
            <FULL STOP character>
            <fileid-characters>1+3
     }
}
```

Figure 49 - Nondirectory file identifierschema

– There shall not be more than one descriptor in a directorywith the same File Identifier.

– The value of the File Link Count field in a File Entryshall not exceed 8.

– No File Entries representing symbolic links shall be recorded.

– The maximum length of a resolved pathname(4/8.7.1) shall not exceed 64.

Note 45

For many systems, there are certain file identifiers which will cause problems during interchange. For maximum interchange, the following file identifiers should not be used

AUX CLOCK$ COM**n** CON LPT**m** NUL PRN

*where **n** is one of the four characters DIGITs ONE to FOUR and **m** is one of the three characters DIGITs ONE to THREE.*

Note 46

The restriction on the maximum size of resolved pathnames may be difficult to enforce incrementally. For example, changing a directory 's name requires, in principle, checking all pathnames including that directory. It may be simpler to check this restriction as a separate processing step prior to interchange

15.2 Level 2

At Level 2, the following restrictionsshall apply:

– The number in any Length of File Identifier and Length of Component Identifier field shall not exceed 14.

– The maximum length of a resolved pathname(4/8.7.1) shall not exceed 1023.

– The number in any File Link Count field in a File Entry shall not exceed 8.

Note 47

This interchange level provides compatibility with ISO/IEC 9945-1 file system restrictions.

15.3 Level 3

At Level 3, no restrictions shall apply.

Section 3 - Requirements for systems for file structure

16 Requirements for the description of systems

Part 4 specifies that certain information shall be communicated between a user and an implementation. Each implementation that conforms to Part 4 shall have a description that identifies the means by which the user may supply or obtain such information.

Note 48

The specifics of the description and the means referred to above will vary from implementation to implementation. For example, an implementation might support two interfaces: a preferred, convenient interface which might vet user input, and a deprecated low level interface which allows any input specified by Part 4.

17 Requirements for an originating system

17.1 General

The implementation shall be capable of recording a set of files, and all descriptors as specified in 4/14, on a volume set according to one of the medium interchange levels specified in 4/15.

The implementation shall be capable of recording a list of character sets (see 1/7.2.11) in which the bit for Character Set Type CS2 shall be set to ONE.

If any information in the scope of a `regid` (1/7.4) is modified and the implementation cannot ensure that the information recorded within the scope of that `regid` still conforms to the agreement implied by the identification in that `regid`, then the implementation shall set the Dirty bit of the Flags field of that `regid` to ONE and should not alter the Identifier field of that `regid` (see 4/17.2.3).

If a domain is identified in a File Set Descriptor and the file set identified is modified and the implementation cannot ensure that the file set still conforms to the agreement implied by the domain identifier, then the implementation shall set the Dirty bit (see 1/7.4) to ONE and may set the Domain Identifier Field to all #00 bytes.

17.2 Mandatory access by user

17.2.1 Files

The implementation shall obtain from the user the information that constitutes the set of files to be recorded.

17.2.2 File set

The implementation shall allow the user to specify which file set to use on a logical volume and to identify the volumes on which the logical volume is recorded.

If the user specifies a logical volume without specifying which file set to use, then the implementation shall use the file set described by the File Set Descriptor having file set number 0.

17.2.3 Descriptors

The implementation shall allow the user to supply the information that is to be recorded in each of the following descriptor fields, and shall supply the information for a field if the user does not supply it.

File Set Descriptor:

- Maximum Interchange Level
- Maximum Character Set List
- File Set Number
- Logical Volume Identifier Character Set

- Logical Volume Identifier
- File Set Identifier
- Copyright File Identifier

File Identifier Descriptor:

- File Version Number
- File Characteristics
- File Identifier

The implementation shall not modify the information that is recorded in each of the following descriptor fields except when directed to do so by the user:

- Maximum Interchange Level field of a File Set Descriptor
- Maximum Character Set List field of a File Set Descriptor
- Except as specified in 4/17.1, Dirty or Protected bits of any regid (1/7.4) field
- Contiguous bit of the Flags field of a File Entry
- Non-relocatable bit of the Flags field of a File Entry
- Existence bit of the File Characteristics field of a directory descriptor

17.3 Optional access by user

If the implementation permits the user to supply the information that is to be recorded in any of the following descriptor fields, the implementation shall record such information as supplied by the user, and shall supply the information for a field if the user does not supply it.

File Set Descriptor:

- Character Set List
- File Set Character Set
- Abstract File Identifier
- Domain Identifier

File Entry:

- Uid
- Gid
- Permissions
- Record Format
- Record Display Attributes
- Record Length
- Information Length
- Access Date and Time
- Modification Date and Time
- Attribute Date and Time
- Checkpoint

Extended Attribute Descriptor:

- Attribute Type
- Attribute Subtype
- Attribute Information

17.3.1 Records

If the implementation allows the user to specify that the information constituting a file is to be interpreted according to 4/14.9.7, the implementation shall obtain from the user the length of each record in the file and the bytes constituting the data space of the file.

17.3.2 File types

If the implementation allows the user to specify that a file is to be interpreted as (see 4/14.6.6) either a block special device file as specified by ISO/IEC 9945-1, or a character special device file as specified by ISO/IEC 9945-1, or a FIFO file as specified by ISO/IEC 9945-1, or according to the `C_ISSOCK` file type identified by ISO/IEC 9945-1, the implementation shall record the attributes supplied by the user for that file and shall not record those attributes if the user does not supply them.

17.3.3 Permissions

The implementation should provide access to files and directories according to either, or both, of 4/14.9.5 or 4/14.10.4. However, as the implementation's security scheme might be incompatible with these schemes, the implementation is not required to provide such access.

17.4 Restrictions

17.4.1 Multivolume volume sets

The implementation shall not be required to record information on the volumes of a volume set that have been assigned a sequence number n, where $1 \leq n < m$, after any information has been recorded on the volume of the volume set that has been assigned sequence number m.

The implementation shall not be required to record information on the volume of a volume set that has been assigned sequence number $m+1$ if there is sufficient space to record the information on the volume that has been assigned a sequence number m, where $1 \leq n \leq m$.

17.4.2 Record length

The implementation may impose a limit on the length of a record that may be recorded in a file. The implementation is not required to record any byte beyond the first m bytes of a record, where m is the value of the imposed limit. The value of m shall be not less than 2048.

17.4.3 File Times

If the File Times Extended Attribute is not recorded for a file, then the implementation shall behave as if the File Times Extended Attribute were recorded with the File Time Existence field having a value of 0.

17.4.4 Information Times

If the Information Times Extended Attribute is not recorded for a file, then the implementation shall behave as if the Information Times Extended Attribute were recorded with the Information Time Existence field having a value of 0.

17.4.5 Alternate Permissions

The implementation may ignore bits 0-3 of the Permissions field of the Alternate Permissions (4/14.10.4) extended attribute.

If requested by the owner of the file, the implementation may ignore bits 4-7 of the Alternate Permissions (4/14.10.4) extended attribute

18 Requirements for a receiving system

18.1 General

The implementation shall be capable of reading the files, and the recorded descriptors as specified in 4/14, from a volume set that has been recorded according to one of the medium interchange levels specified in 4/15.

If the user specifies a logical volume without specifying which file set to use, then the implementation shall use the file set described by the File Set Descriptor having file set number 0.

18.2 Files

The implementation shall make available to the user the information that constitutes the recorded files.

If the implementation allows the user to specify that the information constituting a file is to be interpreted according to 4/14.9.7, the implementation shall make available to the user the length of each record in the file and the display attributes of the file.

18.2.1 File types

If the implementation allows the user to specify that a file is to be interpreted as (see 4/14.6.6) either a block special device file as specified by ISO/IEC 9945-1, or a character special device file as specified by ISO/IEC 9945-1, or a FIFO file as specified by ISO/IEC 9945-1, or according to the `C_ISSOCK` file type identified by ISO/IEC 9945-1, the implementation shall make available to the user the attributes of that file.

18.2.2 Permissions

The implementation should provide access to files and directories according to either, or both, of 4/14.9.5 or 4/14.10.4. However, as the implementation's security scheme might be incompatible with these schemes, the implementation is not required to provide such access.

18.3 Mandatory access by user

The implementation shall allow the user to supply information sufficient to enable the implementation to locate the files required by the user, and to locate the volumes on which these files are recorded.

18.3.1 Descriptors

The implementation shall allow the user to access the information that is recorded in each of the following descriptor fields.

File Set Descriptor:

- Maximum Interchange Level
- Maximum Character Set List
- File Set Identifier
- Copyright File Identifier
- Domain Identifier

File Identifier Descriptor:

- File Version Number
- File Characteristics
- File Identifier

18.4 Restrictions

18.4.1 Record length

The implementation may impose a limit on the length of a record to be made available to the user. The implementation is not required to make available to the user any byte beyond the first *m* bytes of a record, where *m* is the value of the imposed limit. The value of *m* shall be not less than 2048.

18.4.2 File Times

If the File Times Extended Attribute is not recorded for a file, then the implementation shall behave as if the File Times Extended Attribute were recorded with the File Time Existence field having a value of 0.

18.4.3 Information Times

If the Information Times Extended Attribute is not recorded for a file, then the implementation shall behave as if the Information Times Extended Attribute were recorded with the Information Time Existence field having a value of 0.

18.4.4 Alternate Permissions

The implementation may ignore bits 0-3 of the Permissions field of the Alternate Permissions (4/14.10.4) extended attribute.

If requested by the owner of the file, the implementation may ignore bits 4-7 of the Alternate Permissions (4/14.10.4) extended attribute.

Annex A

(normative)

ICB Strategies

A.1 General

This annex specifies four strategies for constructing ICB hierarchies (see 4/8.10.1).

A.2 Strategy 1

This clause specifies a strategy where each ICB of an ICB hierarchy is an extent of k entries, where k is the value of the Maximum Number of Entries field of the ICB Tag field.

The root ICB of the ICB hierarchy shall contain d direct entries, where d is recorded as a `Uint16` (1/7.1.3) in the Strategy Parameter field of the ICB Tag field (see 4/14.6.8), and $i = k - d$ indirect entries. The first indirect entry shall specify the address, referred to as a type I_1 address, of an extent of k direct entries. For $n > 1$, the nth indirect entry shall specify the address, referred to as a type I_n address of an extent of k indirect entries, each of which specifies a type I_{n-1} address.

Note A.1

An example of an ICB hierarchy recorded using this strategy is shown in figure A.1.

The maximum number of direct entries that can be recorded in an ICB hierarchy specified by the strategy of this clause, denoted by *nde(d, i)*, shall be

$$nde(d, i) = d + k + k^2 + ... + k^i$$

$$= d + \frac{k^{i+1} - k}{k - 1}$$

Note A.2

This strategy works well over a large range of ICB hierarchy sizes but works best when there is a single indirect entry in the root ICB, that is, where the actual number of recorded direct entries in an ICB hierarchy is not greater than d+k.

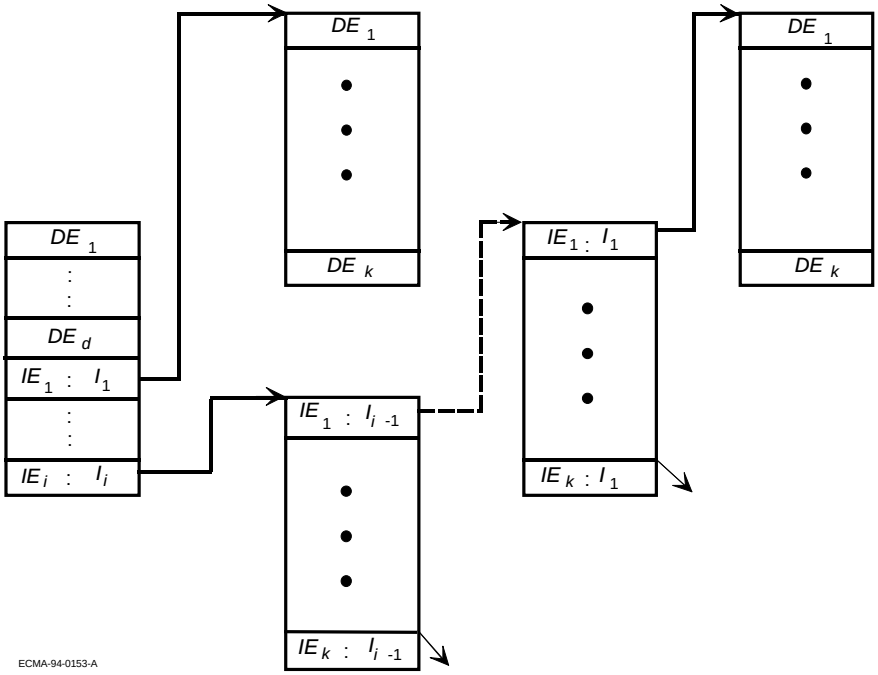

Figure A.1 - Strategy 1 Example

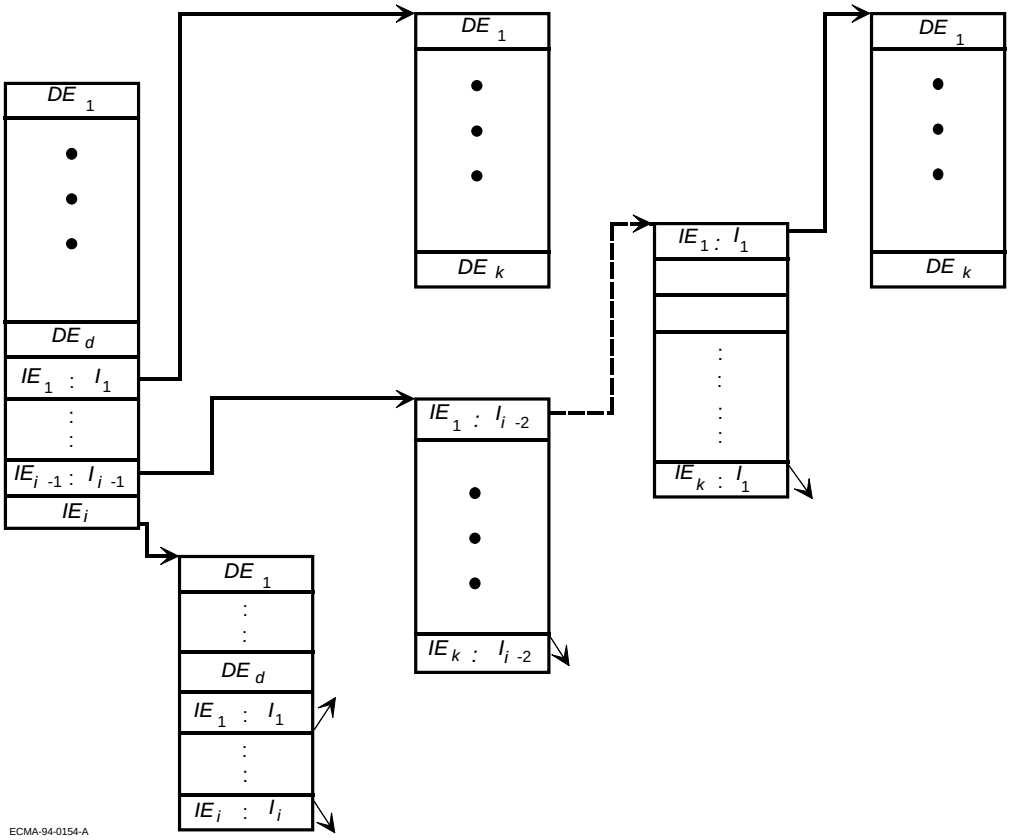

Figure A.2 - Strategy 2 Example

A.3 Strategy 2

This clause specifies a strategy which constructs a list of ICB hierarchies based on the ICB hierarchies constructed with strategy 1.

The root ICB of the ICB hierarchy shall be a master ICB.

A master ICB shall contain d direct entries, where d is recorded as a `Uint16` (1/7.1.3) in the Strategy Parameter field of the ICB Tag field (see 4/14.6.8), and $i = k - d$ indirect entries. The first indirect entry shall specify the address, referred to as a type I_1 address, of an extent of k direct entries. For $1 < n < i$, the nth indirect entry shall specify the address, referred to as a type I_n address of an extent of k indirect entries, each of which specifies a type I_{n-1} address. The ith indirect entry shall specify another master ICB. All master ICBs in the ICB hierarchy shall specify the same values for k and d.

Note A.3

An example of an ICB hierarchy recorded using this strategy is shown in figure 4/A.2.

Note A.4

The number of direct entries that can be recorded in an ICB hierarchy specified by the strategy of this clause is limited only by the size of the logical volume it is recorded on.

A.4 Strategy 3

This clause specifies a strategy where each ICB of the ICB hierarchy is an extent of k entries, where k is the value of the Maximum Number of Entries field of the ICB Tag field, and there are h levels in the ICB hierarchy. The value of h shall be recorded as a `Uint16` (1/7.1.3) in the Strategy Parameter field of the ICB Tag field (see `icbtag` (4/14.6)) for each ICB of the ICB hierarchy. Each ICB at level h of the ICB hierarchy shall consist of k direct entries. For $1 \leq n < h$, each ICB at level n consist of k indirect entries, each of which specifies the address, referred to as I_n, of an ICB at level $n + 1$ of the ICB hierarchy.

The maximum number of direct entries that can be recorded in an ICB hierarchy specified by the strategy of this clause, denoted by *nde(k, h)*, shall be

$nde(k, h) = k^h$

Note A.5

This strategy builds an ICB a hierarchy that is a balanced tree. That is, all the entries at the bottom are direct entries and all the others are indirect entries. An example of an ICB hierarchy recorded using this strategy is shown in figure 4/A.3.

This strategy gives more (total) direct entries in a given ICB hierarchy than strategy 1 for a given d+i. However, the access time is constant and larger than the average access time for strategy 1.

A.5 Strategy 4

For this strategy, the ICB hierarchy shall consist of a single ICB having one direct entry

Note A.6

This strategy works well for rewritable media. The direct entry would be overwritten each time a new direct entry is required for the file described by the ICB.

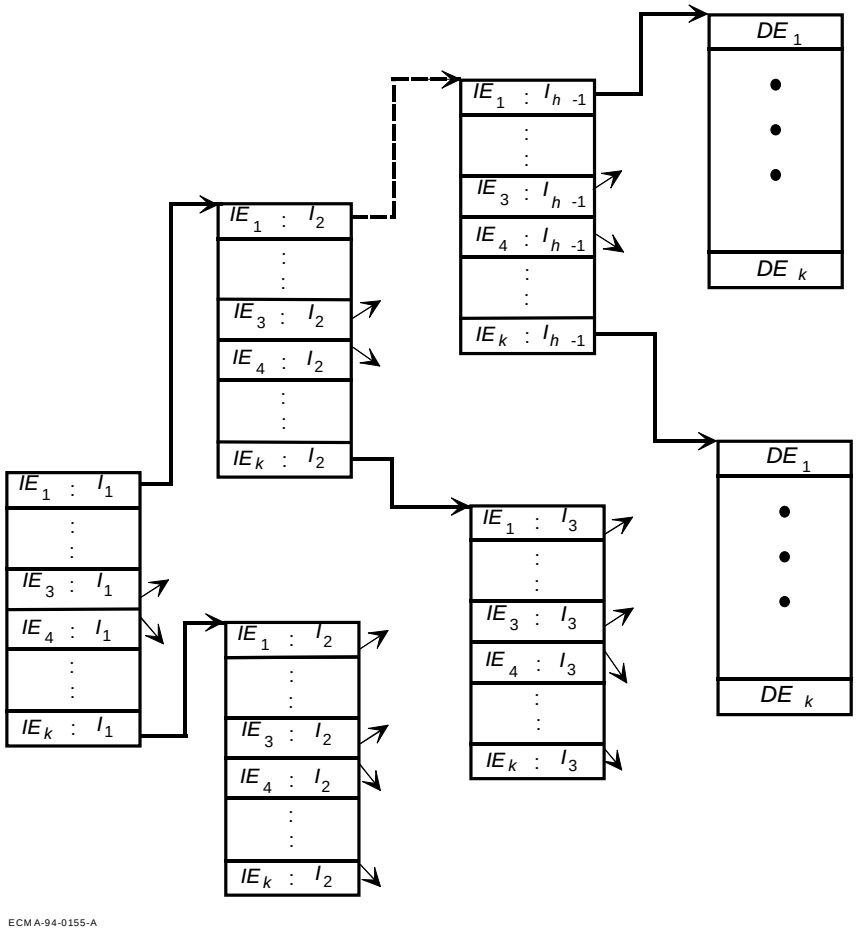

Figure A.3 - Strategy 3 Example

Standard ECMA - 167

Volume and File Structure of Write-Once and Rewritable Media using Non-Sequential Recording for Information Interchange

Part 5: Record structure

Section 1 - General

1 Scope

Part 5 specifies a format and associated system requirements for record structure by specifying:

- record structures intended for use when the information constituting a file is required to be interpreted as a set of records;
- the attributes of the records of a file;
- requirements for the processes which are provided within information processing systems, to enable information to be interchanged between different systems; for this purpose it specifies the functions to be provided within systems which are intended to originate or receive media which conform to Part 5.

2 Parts references

See 1/2.

3 Part interface

This clause specifies the interface of Part 5 to other standards or Parts.

3.1 Input

Part 5 requires the specification of the following by another standard or Part.

- Data space of a file (see 5/6.1).
- If the records of the file are to be interpreted according to 5/9.2.4, 5/9.2.5, 5/9.2.6, 5/9.2.7 or 5/9.2.8 or are intended to be displayed according to 5/9.3, specification of how characters, including the LINE FEED, VERTICAL TABULATION, FORM FEED, and CARRIAGE RETURN characters, are encoded within the data space of the file.

3.2 Output

Part 5 specifies the following which may be used by other standards or Parts.

- Identification and specification of record types (see 5/9.2).
- Identification and specification of record display attributes (see 5/9.3).

4 Reference

ISO/IEC 1539:1991, *Information technology - Programming languages - FORTRAN*

5 Conformance

See 1/3.

6 Definitions

In addition to the definitions of Part 1 (see 1/5), the following definition applies for Part 5.

6.1 Data space of a file

The set of bytes specified for a file shall be the data space of the file.

The bytes of the set shall be numbered with consecutive integers assigned in an ascending sequence. The numbering shall start from 0 which shall be assigned to the first, if any, byte of the file.

7 Notation

The notation of Part 1 (see 1/6) applies to Part 5.

8 Basic types

In addition to the basic types of Part 1 (see 1/7), the following basic type applies to Part 5.

8.1 16-bit unsigned numerical values (MSB)

A `Uint16MSB` value, represented by the hexadecimal representation #wxyz, shall be recorded in a two-byte field as #wx #yz.

Note 1

For example, the decimal number 4 660 has #1234 as its hexadecimal representation and shall be recorded as #12 #34.

Section 2 - Requirements for the medium for record structure

9 Record structure

The information in a file may be organised as a set of records (see 1/5.8) according to Part 5. The length of a record shall be the number of bytes in the record. A record shall be recorded in a container which shall be recorded in the data space of a file. This container shall be referred to as a Measured Data Unit (MDU) (see 5/9.1).

9.1 Relationship to a file

Each MDU shall comprise a set of successive bytes of the data space of the file (see 5/6.1). The first or only MDU shall begin at the first byte of the data space of the file. Each successive MDU shall begin at the byte of the data space of the file immediately following the last byte of the preceding MDU.

If there are no bytes in the data space of the file, then no MDU shall be considered to have been recorded in the file.

9.2 Record type

A record of a file recorded according to Part 5 shall be one of the following types:

- padded fixed-length (5/9.2.1)
- fixed-length (5/9.2.2)
- variable-length-8 (5/9.2.3.1)
- variable-length-16 (5/9.2.3.2)
- variable-length-16-MSB (5/9.2.3.3)
- variable-length-32 (5/9.2.3.4)
- stream-print (5/9.2.4)
- stream-LF (5/9.2.5)
- stream-CR (5/9.2.6)
- stream-CRLF (5/9.2.7)
- stream-LFCR (5/9.2.8)

All records in a file shall be of the same type.

9.2.1 Padded fixed-length records

A padded fixed-length record shall be a record contained in a file that is assigned to contain records that shall have the same length. The minimum assigned length of a padded fixed-length record shall be 1.

An MDU containing a padded fixed-length record shall be recorded according to the schema shown in figure 5/1.

```
[MDU ]{
      <record>
      <#00 byte> 0+1
}
```

Figure 1 - Padded fixed-length record schema

The #00 byte shall be recorded only if necessary to give the MDU an even length.

9.2.2 Fixed-length records

A fixed-length record shall be a record contained in a file that is assigned to contain records that shall have the same length. The minimum assigned length of a fixed-length record shall be 1.

An MDU containing a fixed-length record shall be recorded according to the schema shown in figure 5/2.

```
[MDU ]{
      <record>
}
```

Figure 2 - Fixed-length recordschema

9.2.3 Variable-length records

A variable-length record shall be a record contained in a file that is assigned to contain records that may have different lengths.

A variable-length record shall be one of the following types:

- variable-length-8 (5/9.2.3.1)
- variable-length-16 (5/9.2.3.2)
- variable-length-16-MSB (5/9.2.3.3)
- variable-length-32 (5/9.2.3.4)

A maximum record length shall be assigned for a file. The length of any record in the file shall not exceed this value. The minimum length of a variable-length record shall be 0.

The length of a variable-length record shall be recorded in a Record Control Word (RCW). The length of a record does not include the size of the RCW. The interpretation of the value of the RCW shall be as given in figure 5/3, where *n* denotes the number of bits in the RCW of a record for the file:

RCW	Interpretation
$2^n - 1$	The RCW is the final RCW of the logical blockin which the RCW is recorded.
0 to $2^n - 2$	The RCW specifies the length of the record

Figure 3 - RCW interpretation

Note 2

The length of the RCWis not included in the number recorded in the RCW.

9.2.3.1 Variable-length-8

An MDU containing a variable-length-8 record shall be recorded according to the schema shown in figure 5/4 where the RCW is recorded as an `Uint8` (1/7.1.1).

```
[MDU ]{
    <RCW>
    {
       <record>
    } 0+1
}
```

Figure 4 - Variable-length-8 recordschema

9.2.3.2 Variable-length-16

An MDU containing a variable-length-16 record shall be recorded according to the schema shown in figure 5/5 where the RCW is recorded as an `Uint16` (1/7.1.3).

```
[MDU ]{
    <RCW>
    {
       <record>
       <#00 byte> 0+1
    } 0+1
}
```

Figure 5 - Variable-length-16 record schema

The #00 byte shall be recorded only if necessary to give the MDU an even length.

9.2.3.3 Variable-length-16-MSB

An MDU containing a variable-length-16-MSB record shall be recorded according to the schema shown in figure 5/6 where the RCW is recorded as an `Uint16MSB` (5/8.1).

```
[MDU ]{
    <RCW>
    {
       <record>
       <#00 byte> 0+1
    } 0+1
}
```

Figure 6 - Variable-length-16-MSB record schema

The #00 byte shall be recorded only if necessary to give the MDU an even length.

Note 3

The use of variable-length-16-MSB records is included only for compatibility with ECMA-119. It is recommended that variable-length-16 records be used instead.

9.2.3.4 Variable-length-32

An MDU containing a variable-length-32 record shall be recorded according to the schema shown in figure 5/7 where the RCW is recorded as an `Uint32` (1/7.1.5).

```
[MDU ]{
    <RCW>
    {
       <record>
    } 0+1
}
```

Figure 7 - Variable-length-32 record schema

9.2.4 Stream-print records

A stream-print record shall be a record contained in a file that is assigned to contain records that may have different lengths.

A maximum record length shall be assigned for a file assigned to contain stream-print records. The length of any record in the file shall not exceed this value. The minimum length of a stream-print record shall be 0.

The first byte of a stream-print record shall not be a #00 byte.

An MDU containing a stream-print record shall be recorded according to the schema shown in figure 5/8.

```
[MDU ]{
      <#00 byte> 0+
      {
            <record> <LINE FEED  character>
         |  <record> <VERTICAL TABULATION  character>
         |  <record> <FORM FEED character>
         |  <record> <CARRIAGE RETURN  character> <LINE FEED  character>
      }
}
```

Figure 8 - Stream-print record schema

9.2.5 Stream-LF records

A stream-LF record shall be a record contained in a file that is assigned to contain records that may have different lengths.

A maximum record length shall be assigned for a file assigned to contain stream-LF records. The length of any record in the file shall not exceed this value. The minimum length of a stream-LF record shall be 0.

An MDU containing a stream-LF record shall be recorded according to the schema shown in figure 5/9.

```
[MDU ]{
      <record> <LINE FEED  character>
}
```

Figure 9 - Stream-LF record schema

9.2.6 Stream-CR records

A stream-CR record shall be a record contained in a file that is assigned to contain records that may have different lengths.

A maximum record length shall be assigned for a file assigned to contain stream-CR records. The length of any record in the file shall not exceed this value. The minimum length of a stream-CR record shall be 0.

An MDU containing a stream-CR record shall be recorded according to the schema shown in figure 5/10.

```
[MDU ]{
      <record> <CARRIAGE RETURN  character>
}
```

Figure 10 - Stream-CR record schema

9.2.7 Stream-CRLF records

A stream-CRLF record shall be a record contained in a file that is assigned to contain records that may have different lengths.

A maximum record length shall be assigned for a file assigned to contain stream-CRLF records. The length of any record in the file shall not exceed this value. The minimum length of a stream-CRLF record shall be 0.

An MDU containing a stream-CRLF record shall be recorded according to the schema shown in figure 5/11.

```
[MDU ]{
      <record> <CARRIAGE RETURN character> <LINE FEED character>
}
```

Figure 11 - Stream-CRLF record schema

9.2.8 Stream-LFCR records

A stream-LFCR record shall be a record contained in a file that is assigned to contain records that may have different lengths.

A maximum record length shall be assigned for a file assigned to contain stream-LFCR records. The length of any record in the file shall not exceed this value. The minimum length of a stream-LFCR record shall be 0.

An MDU containing a stream-LFCR record shall be recorded according to the schema shown in figure 5/12.

```
[MDU ]{
      <record> <LINE FEED  character> <CARRIAGE RETURN  character>
}
```

Figure 12 - Stream-LFCR record schema

9.3 Record display attributes

This clause specifies the processing of the records in a file when they are displayed on a character-imaging device. If the file is not recorded with any of the record types (see 5/9.2) specified in Part 5, then the records of the file need not be processed according to the record display attributes specified by this clause.

A file recorded with records according to Part 5 shall be assigned one of the following types of record display attributes

- LF-CR (5/9.3.1)
- first byte position (5/9.3.2)
- implied (5/9.3.3)

9.3.1 LF-CR display attribute

When displayed on a character-imaging device, each record of the file shall be preceded by a LINE FEED character and followed by a CARRIAGE RETURN character.

9.3.2 First byte position display attribute

When displayed on a character-imaging device, the first byte of each record of the file shall be interpreted as specified in ISO 1539 for vertical spacing.

9.3.3 Implied display attribute

When displayed on a character-imaging device, each record of the file shall be interpreted as containing the necessary control information for the imaging device.

Section 3 - Requirements for systems for record structure

10 Requirements for the description of systems

Part 5 specifies that certain information shall be communicated between a user and an implementation. Each implementation that conforms to Part 5 shall have a description that identifies the means by which the user may supply or obtain such information.

Note 4

The specifics of the description and the means referred to above will vary from implementation to implementation. For example, an implementation might support two interfaces: a preferred, convenient interface which might vet user input, and a deprecated low level interface which allows any input specified by Part 5.

11 Requirements for an originating system

11.1 General

11.1.1 Files

If the implementation allows the user to specify that the information constituting a file is to be interpreted according to a record type specified in 5/9.2, the implementation shall obtain from the user the length of each record in the file.

If the implementation allows the user to specify that the information constituting a file is to be interpreted according to a record display attribute specified in 5/9.3, the implementation shall obtain from the user the record display attribute for the file.

11.1.2 Record length

The implementation may impose a limit on the length of a record that may be recorded in a file. The implementation is not required to record any byte beyond the first *m* bytes of a record, where *m* is the value of the imposed limit. The value of *m* shall be not less than 2048.

12 Requirements for a receiving system

12.1 General

12.1.1 Files

If the implementation allows the user to specify that the information constituting a file is to be interpreted according to a record type specified in 5/9.2, the implementation shall make available to the user the length of each record in the file.

If the implementation allows the user to specify that the information constituting a file is to be interpreted according to a record display attribute specified in 5/9.3, the implementation shall make available to the user the record display attribute for the file.

12.1.2 Record length

The implementation may impose a limit on the length of a record to be made available to the user. The implementation is not required to make available to the user any byte beyond the first *m* bytes of a record, where *m* is the value of the imposed limit. The value of *m* shall be not less than 2048.

Printed copies can be ordered from:

ECMA
114 Rue du Rhône
CH-1204 Geneva
Switzerland

Fax: +41 22 849.60.01
Internet: helpdesk@ecma.ch

Files can be downloaded from our FTP site, **ftp.ecma.ch,** logging in as **anonymous** and giving your E-mail address as **password**. This Standard is available from library **ECMA-ST** as a compacted, self-expanding file in MSWord 6.0 format (file {E167-DOC.EXE) and as a compacted, self-expanding PostScript file (file E167-PSC.EXE). File E167-EXP.TXT gives a short presentation of the Standard.

The ECMA site can be reached also via a modem. The phone number is +41 22 735.33.29, modem settings are 8/n/1. Telnet (at ftp.ecma.ch) can also be used.

Our web site, http://www.ecma.ch, gives full information on ECMA, ECMA activities, ECMA Standards and Technical Reports.

ECMA
114 Rue du Rhône
CH-1204 Geneva
Switzerland

Fax: **+41 22 849.60.01**
Internet: **helpdesk@ecma.ch**

This Standard ECMA-167 is available free of charge in printed form (and as a file).

See inside cover page for ordering instructions.