Standard ECMA-168
2nd Edition - December 1994

ECMA

Standardizing Information and Communication Systems

Volume and File Structure of Read-Only and Write-Once Compact Disk Media for Information Interchange

Standard ECMA-168

2nd Edition - December 1994

ECMA

Standardizing Information and Communication Systems

Volume and File Structure of Read-Only and Write-Once Compact Disk Media for Information Interchange

Part 1 - General

Part 2 - Volume and Boot Block Recognition

Part 3 - Volume and File Structure

Part 4 - Record Structure

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

GL ECMA-168.DOC 17-05-96 10,34

Brief History

The bulk of the work leading to this ECMA Standard was done by the Frankfurt Group, which was formed in 1990 by many CD-ROM and CD-WO hardware and media manufacturers, CD-ROM data publishers, users of CD-ROMs, and major computer companies, to exploit the CD-ROM and CD-Recordable technology.

This ECMA Standard can be used for both CD-ROM and CD-WO media for interchanging files. This ECMA Standard is seen as a revision and extension of ECMA-119 for CD-ROM applications because it has eliminated several restrictions and performance problems of ECMA-119.

During the work in ECMA/TC15, a lot of effort was spent in order to harmonize this ECMA Standard with ECMA-167. This ECMA Standard follows the ECMA-167 volume and file structure framework, and has common definitions with ECMA-167 regarding file attributes and record structure.

The 2nd Edition of this ECMA Standard has been fully aligned with ISO/IEC 13346 and ISO/IEC 13490 which have resulted from the 1st Editions of ECMA-167 and ECMA-168, resp., by their adoption by ISO/IEC JTC 1 under the fast-track procedure.

This ECMA Standard has been adopted as 2nd Edition by the ECMA General Assembly of 15th December 1994.

Table of contents

STANDARD ECMA - 168

Volume and File Structure for Read-Only and Write-Once Compact Disk Media 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 between users of information processing systems using CD-WO (a write-once compact disk medium), hybrid CD-WO (a write-once compact disk with a read-only area) and CD-ROM disks.

NOTE

CD-WO is an evolution of CD-ROM technology which allows the recording of information on a write-once compact disk medium.

NOTE

A volume set may be recorded that is in conformance with both ECMA-119 and ECMA-168. ECMA-168 is an enhancement of ECMA-119. ECMA-168 allows greater information interchange using CD-ROM. In addition, it supports incremental recording and updating of information stored on a CD-WO disk. Under certain restrictions (see 3/B.2.1), all of the files may be read by both a receiving system conforming to ECMA-119 and by a receiving system conforming to ECMA-168.

This ECMA Standard consists of the following four Parts:

Part 1 : General
Part 2 : Volume and Boot Block Recognition
Part 3 : Volume and File Structure
Part 4 : Record Structure
Annex A - Restrictions on a standard for recording, is part of Part 3
Annex B - Methods of interchange, is part of Part 3
Annex C - CD-WO disk format and system requirements, is part of Part 3
Annex D - CD-WO subsystem interface requirements, is part of Part 3

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

2 Parts references

The first digit of a reference in this ECMA Standard identifies the Part. For example, 2/5 refers to clause 5 in Part 2. If the reference is preceded by “figure”, the reference is to a figure. For example, figure 2/5 refers to figure 5 in Part 2. If the reference is preceded by “table”, the reference is to a table. For example, table 2/5 refers to table 5 in Part 2.

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 1/5.13) and all information recorded on it conforms to the specifications of all Parts, or to Parts 1, 2 and 3. A statement of conformance shall identify the Parts, and the levels of medium interchange (see 2/10 and 3/16) to which the contents 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 all Parts or in Parts 1, 2 and 3 either for an originating system (see 2/12, 3/18 and 4/11) or for a receiving system (see 2/13, 3/19 and 4/12) or for both types of system. A statement of conformance shall identify the Parts and the levels of the requirements for the 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 ets (1991)
ECMA-94	8-Bit Single-Byte Coded Graphic Character Sets - Latin Alphabets No. 1 to No. 4 (1986)

ECMA-119	Volume and File Structure of CD-ROM for Information Interchange (1987)
ECMA-130	Data Interchange on Read-Only 120 mm Optical Data Disks (CD-ROM) (1996)
ECMA-167	Volume and File Structure for Write-Once and Rewritable Media using Non-Sequential Recording for Information Interchange (1994)

ISO/IEC 9945-1:1990, Information technology — Portable Operating System Interface (POSIX) — Part 1: System Application Program Interface (API) [C Language].

ISO/IEC 10646-1:1993, Information technology — Universal Multiple-Octet Coded Character Set (UCS) — Part 1: Architecture and Basic Multilingual Plane.

ISO/IEC 13800, Information technology – Procedure for the registration of identifiers and attributes for volume and file structure

IEC 908:1987, Compact disc digital audio system.

5 Definitions

For the purposes 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 1/5.13) specifies that the container for the recording of a byte has more than eight bits, 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 CD-ROM

A read-only compact disk (see 1/5.5).

5.4 CD-WO

A write-once compact disk (see 1/5.5) that conforms to a standard for recording (see 1/5.13).

5.5 compact disk

An optical disk that is recorded according to IEC 908 and ECMA-130.

5.6 descriptor

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

5.7 file

A collection of information.

5-8 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.

5.9 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.

5.10 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.

5.11 record

A sequence of bytes treated as a unit of information.

5.12 sector

The data field of an addressable part of the medium that can be accessed independently of other addressable parts of the medium as specified in the standard for recording (see 1/5.13).

5.13 standard for recording

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

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

5.14 user

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

5.15 volume

A sector address space as specified in the standard for recording (see 1/5.13).

NOTE

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.

5.16 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. Numerical notation

6.1.1 Decimal notation

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 table specifying the location, length, name and contents of each field. The interpretation of each field shall be given in the prose associated with the table.

Table 1 - Example descriptor format

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` (1/7.1.4) =57
42	D_L	Implementation Use	bytes
[D_L+42]	*	Padding	#00 bytes

The descriptor specified by table 1/1 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 [41+D_L], 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 for 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*.

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 table 1/2.

Table 2 - Repetition operators

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

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 1/1specifies 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 1 - Example schema

6.7 Other notations

Other notations used in this ECMA Standard are specified in table 1/3.

Table 3 - Other notation

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

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 $0 \leq x < 2^n$, 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

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

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

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

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

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` (see 1/7.2.12) or d-characters. The specification of the 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

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, 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 table 1/4.

Table 4 - `charspec` format

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

7.2.1.1 Character Set Type (RBP 0)

This field shall specify the allowed characters by identifying a set of characters shown in table 1/5.

Table 5 - Sets of allowed characters

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.

NOTE

Briefly, these character sets are:

CS0 – by agreement

CS1 – the whole or any subset of the graphic characters specified by ISO/IEC 10646-1

CS2 – 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 – the 63 graphic characters of the portable ISO/IEC 9945-1 file name set

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

CS5 – the 191 graphic characters of ECMA-94

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

CS7 – 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 – 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

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

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

These characters are: HYPHEN-MINUS, FULL STOP, DIGITs, LATIN CAPITAL LETTERs, LATIN SMALL 7.2.

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, Part 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

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` (see 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` (see 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 (see 1/7.2) are recorded. The number of bytes used to record the characters shall be recorded as a `Uint8` (see 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 format

A timestamp shall be recorded either as a `timestamp` (see 1/7.3.1) or as a short`timestamp` (see 1/7.3.2).

7.3.1 Timestamp

A `timestamp` shall specify a date and time recorded in the format shown in table 1/6. If all fields are 0, it shall mean that the date and time are not specified.

Table 6 - `timestamp` format

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)

7.3.1.1 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 table 1/7. 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 of the number is -2 047, then no such offset is specified.

Table 7 - `timestamp` interpretation

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.

7.3.1.2 Year (RBP 2)

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

7.3.1.3 Month (RBP 4)

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

7.3.1.4 Day (RBP 5)

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

7.3.1.5 Hour (RBP 6)

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

7.3.1.6 Minute (RBP 7)

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

7.3.1.7 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.

7.3.1.8 Centiseconds (RBP 9)

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

7.3.1.9 Hundreds of Microseconds (RBP 10)

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

7.3.1.10 Microseconds (RBP 11)

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

7.3.2 Short timestamp

A `shorttimestamp` shall specify a date and time recorded in the format shown in table 1/8. If all fields are 0, it shall mean that the date and time are not specified.

NOTE

The `shorttimestamp` is defined for use in the Directory Record (see 3/15.1) so as to be compatible with the Directory Record specified in ECMA-119.

Table 8 - `shorttimestamp` format

RBP	Length	Name	Contents
0	1	Year	`Uint8` (1/7.1.1)
1	1	Month	`Uint8` (1/7.1.1)
2	1	Day	`Uint8` (1/7.1.1)
3	1	Hour	`Uint8` (1/7.1.1)
4	1	Minute	`Uint8` (1/7.1.1)
5	1	Second	`Uint8` (1/7.1.1)
6	1	Time Zone	`Int8` (1/7.1.2)

7.3.2.1 Year (RBP 0)

This field shall specify the number of years since 1900.

7.3.2.2 Month (RBP 1)

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

7.3.2.3 Day (RBP 2)

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

7.3.2.4 Hour (RBP 3)

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

7.3.2.5 Minute (RBP 4)

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

7.3.2.6 Second (RBP 5)

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

7.3.2.7 Time Zone (RBP 6)

This field shall specify the offset from Coordinated Universal Time, in 15 minute intervals, as a number in the range -48 (West) to +52 (East).

7.4 Entity Identifier

A `regid` specifies an entity identification and shall be recorded in the format shown in table 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`.

Table 9 - `regid` format

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

7.4.1 Flags (RBP 0)

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

Table 10 - Characteristics of `regid`

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 the `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 3/18.1).
2-7	shall be reserved for future standardisation and all bits shall be set to ZERO.

7.4.2 Identifier (RBP 1)

If the first byte of this field contains #2B, then this field contains an identifier specified by 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

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 - 168

Volume and File Structure for Read-Only and Write-Once Compact Disk Media 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 Cross-reference

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 1/5.13).
- 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 1/5.13). 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

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 2/2.

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 2/3.

Identifier	Interpretation
“BEA01”	According to 2/9.2.
“BOOT2”	According to 2/9.4.
“CD001”	According to ECMA-119.
“CDW02”	According to this ECMA Standard.
“NSR02”	According to 3/9.1 of ECMA-167.
“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 Descriptor shall be recorded in the format shown in figure 2/4.

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 2/5.

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 2/6.

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

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

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

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 - 168

Volume and File Structure for Read-Only and Write-Once Compact Disk Media for Information Interchange

Part 3 : Volume and File structure

Section 1 - General

1 Scope

Part 3 specifies a format and associated system requirements for volume and file 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 placement of files;
- the attributes of the files;
- the relationship among files of a file set;
- the relationship among file sets of a volume set;
- 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 3 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 Cross - reference

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 1/5.3).
- A volume set of one or more volumes (see 3/9.2).
- For the purposes of ECMA-167, Part 2, a volume recognition sequence (see 2/8.3.1) shall be recorded as specified in 3/9.1.7.
- For the purposes of ECMA-167, Part 2, the volume recognition space (see 2/8.2) shall be as specified in 3/9.1.6.
- For the purposes of ECMA-167, Part 2, the initial sector (see 2/3.1) of the volume shall be sector number 0 of the volume as specified in 3/9.1.1.1.

3.2 Output

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

- A volume space for a volume (see 3/9.1.4).
- A volume set of one or more volumes (see 3/9.2).
- An indication that a volume may have been recorded according to this Part (see 3/9.1.7)
- Volume partitions (see 3/9.1.4.3).
- Sessions (see 3/9.1.3).
- Logical blocks of a fixed size for a volume set (see 3/9.1.4.1).
- The size of a logical block.
- Attributes of a volume.
- Attributes of a volume partition.
- Data space of a file (see 3/13.5.2).
- Attributes of a file.
- Attributes of a directory.

– Attributes of a directory hierarchy.

4 Conformance

4.1 Conformance of a medium

A medium shall be in conformance with this ECMA Standard when it conforms to a standard for recording (see 1/5.13) and all information recorded on it conforms to the specifications of all Parts, or to Parts 1, 2 and 3. A statement of conformance shall identify the Parts, and the levels of medium interchange (see 2/10 and 3/16) to which the contents of the medium conform.

4.2 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 all Parts or in Parts 1, 2 and 3 either for an originating system (see 2/12, 3/18 and 4/11) or for a receiving system (see 2/13, 3/19 and 4/12) or for both types of system. A statement of conformance shall identify the Parts and the levels of the requirements for the Parts which can be met by the system.

5 References

ECMA-35	Code Extension Techniques (1994)
ECMA-48	Control Functions for Coded Character Sets (1991)
ECMA-119	Volume and File Structure of CDROM for Information Interchange (1987)
ECMA-130	Data Interchange on Read-Only 120 mm Optical Data Disks (CD-ROM) (1996)
ECMA-167	Volume and File Structure for Write-Once and Rewritable Media using Non-Sequential Recording for Information Interchange (1994)

ISO/IEC 9945-1:1990, Information technology — Portable Operating System Interface (POSIX) — Part 1: System Application Program Interface (API) [C Language].

ISO/IEC 10646-1:1993, Information technology — Universal Multiple-Octet Coded Character Set (UCS) — Part 1: Architecture and Basic Multilingual Plane.

ISO/IEC 13800, Information technology – Procedure for the registration of identifiers and attributes for volume and file structure

IEC 908:1987, Compact disc digital audio system.

6 Definitions

In addition to the definitions of Part 1, the following definitions apply.

6.1 file set

A collection of files and directories.

6.2 group ID

An identification of a group of users.

6.3 logical block

The unit of allocation of a volume set.

6.4 user ID

An identification of a user.

7 Notation

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

8 Basic types

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

8.1 16-Bit unsigned numerical values with most significant byte first

A `Uint16MSB` value, represented by the hexadecimal representation #wxyz, shall be recorded in a two-byte field as #wx #yz.

NOTE

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

8.2 16-Bit signed numerical values with most significant byte first

An `Int16MSB` value, represented in two's complement form by the hexadecimal representation #wxyz, shall be recorded in a two-byte field as #wx #yz.

NOTE

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

8.3 16-Bit unsigned numerical values with both byte orders

A `Uint16BOTH` value, represented by the hexadecimal representation #wxyz, shall be recorded in a four-byte field as #yz #wx #wx #yz.

NOTE

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

8.4 32-Bit unsigned numerical values with most significant byte first

A `Uint32MSB` value, represented by the hexadecimal representation #stuvwxyz, shall be recorded in a four-byte field as #st #uv #wx #yz.

NOTE

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

8.5 32-Bit signed numerical values with most significant byte first

An `Int32MSB` value, represented in two's complement form by the hexadecimal representation #stuvwxyz, shall be recorded in a four-byte field as #st #uv #wx #yz.

NOTE

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

8.6 32-Bit unsigned numerical values with both byte orders

A `Uint32BOTH` value, represented by the hexadecimal representation #stuvwxyz, shall be recorded in a eight-byte field as #yz #wx #uv #st #st #uv #wx #yz.

NOTE

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

8.7 Volume structure descriptor format

Volume structure descriptors specified in this Part shall be recorded in the format shown in table 3/1.

Table 1 - CD–WO Volume structure descriptor format

BP	Length	Name	Contents
0	1	Structure Type	`Uint8` (1/7.1.1)
1	5	Standard Identifier	bytes = “CDW02”
6	1	Structure Version	`Uint8` (1/7.1.1) = 2
7	2 041	Structure Data	bytes

8.7.1 Structure Type (BP 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 this Part. Types 1-6 and 255 are specified as shown in table 3/2. All other types are reserved for future standardisation by this Part.

Table 2 - Structure Type Interpretation

Type	Identification
1	Primary Volume Descriptor (see 3/10.1)
2	Supplementary Volume Descriptor (see 3/10.2)
3	File Set Descriptor (see 3/12.1)
4	Implementation Use Descriptor (see 3/12.2).
5	Volume Partition Descriptor (see 3/10.3)
6	End Transaction Descriptor (see 3/01.4)
255	Terminating Descriptor (see 3/10.5)

8.7.2 Standard Identifier (BP 1)

This field shall specify “CDW02”.

8.7.3 Structure Version (BP 6)

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

8.7.4 Structure Data (BP 7)

The interpretation of this field shall be specified by the clause identified by the Structure Type field (see 3/8.7.1).

Section 2 - Requirements for the medium for volume and file structure

9 Volume structure

9.1 Arrangement of information on a volume

9.1.1 Sectors

9.1.1.1 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 standard for recording (see 1/5.13). Sector number 0 shall be assigned to the sector having the lowest physical address of the volume as specified in the standard for recording.

9.1.1.2 Data space of a sector

The first 2^n bytes of a sector, where n is the largest integer greater than 0 such that 2^n is less than or equal to the number of bytes in the sector (see 1/5.12), shall be referred to as the data space of the sector. The length of the data space of a sector shall not be less than 2 048 bytes. The interpretation of the bytes, if any, in the sector after the data space is not specified by this Part.

9.1.1.3 Data sector

A data sector shall be a sector that is specified in the standard for recording as available for recording information specified by this Part.

9.1.1.4 Unrecorded sectors

A sector is unrecorded if the standard for recording allows detection that a sector has been unrecorded and if that sector is unrecorded. Any unrecorded sector shall be interpreted as containing all #00 bytes.

9.1.2 Track

The sectors of a volume shall be organised into one or more tracks. A track shall be a sequence of one or more sectors, the sector numbers of which form a continuous ascending sequence. No sector shall belong to more than one track.

NOTE

There may be gaps between tracks; that is, the last sector of a track need not be adjacent to the first sector of the next track.

9.1.2.1 Track number

Each track of a volume shall be identified by a unique number. Track numbers shall be consecutive integers assigned in an ascending sequence starting with 1. Track number 1 shall be assigned to the track having sector number 0 as the first, or only, constituent sector. The numbering shall continue through successive tracks, each of which begins with a sector having a greater sector number than that of the last sector constituting the previous track of the volume. The track having the greatest assigned track number of a volume shall be the last track of the volume.

9.1.2.2 Data track

Data tracks are tracks that are designated to contain data sectors having the attributes specified in 3/A.1.

9.1.2.2.1 Recording mode of a data track

A data track shall be assigned to be recorded in either incremental mode (see 3/A.2) or track-at-once mode (see 3/A.3). A data track assigned to be recorded in incremental mode shall be assigned to contain either a sequence of variable-length packets (see 3/9.1.2.2.1.2), or a sequence of fixed-length packets (see 3/9.1.2.2.1.1).

A packet shall be a sequence of one or more data sectors, the sector numbers of which form a continuous ascending sequence. The length of a packet shall be the number of data sectors in the packet, and each packet shall be treated as if it is recorded as a unit.

9.1.2.2.1.1 Fixed-length packet

A fixed-length packet shall be a packet recorded in a data track assigned to contain packets that shall have the same length.

The sector number of the first data sector in a fixed-length packet shall be one greater than the sector number of the last data sector in the previous packet, if any, in the same track. The sector number of the first data sector in the first fixed-length packet in a data track shall be the sector number of the first data sector in the track.

9.1.2.2.1.2 Variable-length packet

A variable-length packet shall be a packet recorded in a data track assigned to contain packets that may have different lengths.

The sector number of the first data sector in a variable-length packet shall be *n* greater than the sector number of the last data sector in the previous packet in the same track, if any. The sector number of the first data sector in the first variable-length packet in a data track shall be *n* greater than the sector number of the first data sector in the track. The value *n* is specified in the standard for recording.

9.1.3 Session

The tracks of a volume shall be organised into one or more sessions as specified by the standard for recording. A session shall be a sequence of one or more tracks, the track numbers of which form a continuous ascending sequence.

Each session of a volume shall have an assigned session number. Session numbers shall be consecutive integers assigned in an ascending sequence starting with 1. Session number 1 shall be assigned to the session having track number 1 as the first, or only, constituent track. The numbering shall continue through successive sessions, each of which begins with the track having the next greater track number than that of the last track constituting the previous session of the volume.

The session having the greatest assigned session number of a volume shall be the last session of the volume.

9.1.4 Volume space

The set of all sectors in a volume whose sector numbers are less than or equal to the sector number of the last sector of the last track of the volume shall be referred to as the volume space of the volume. The information on a volume shall be recorded in the set of all data sectors of the volume space. The bytes in the volume space shall be numbered with consecutive integers assigned in an ascending sequence starting with 0. Let *s* be the number of bytes in the data space of a sector; then byte *b* of the volume space is byte *rem(b, s)* of sector *ip(b/s)*.

NOTE

Sectors not identified as being data sectors (see 3/9.1.1.3) are nevertheless part of the volume space. Thus, not necessarily all of the volume space can be used for recording information specified by this Part.

9.1.4.1 Logical block

The volume space 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 2^n bytes, where *n* is an integer greater than or equal to 9, such that the logical block size is not greater than the length of the data space of any sector in any volume of the volume set (see 3/9.2). The logical blocks in the volume space shall be numbered with consecutive integers assigned in an ascending sequence starting with 0. Let *lbs* be the number of bytes in a logical block, then logical block number *lbn* of the volume space begins at byte $lbn \times lbs$ of the volume space. The logical block size of each volume in a volume set shall be the same.

NOTE

A logical block ends in the same sector in which it begins.

9.1.4.2 Extent

An extent shall be 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 address of the first logical block in the sequence. An extent shall end in the same track in which it begins.

9.1.4.3 Volume partition

A volume partition shall be recorded over an extent. The first logical block of the extent shall have a logical block number which is the lowest logical block number in the sector that contains the logical block. The identification and attributes of a volume partition shall be recorded in a Volume Partition Descriptor (see 3/10.3). The interpretation of the contents of a volume partition is not specified by this Part.

NOTE

Volume 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 volume partitions to use.

9.1.5 Unallocated space of a volume

Sectors whose sector numbers are greater than the sector number of the last sector of the last track of a volume but less than, or equal to, the maximum sector number of the volume allowed by the standard for recording are referred to as unallocated sectors. Unallocated sectors do not belong to the volume space of a volume.

If allowed by the standard for recording, the unallocated sectors may be later added to the volume space of a volume by allocating them to any new track added to the volume.

9.1.6 Volume recognition space

The volume recognition space (see 2/8.2) of a volume shall be the part of the volume space starting at the first byte of the first data sector of the first recorded data track in the last session of the volume and ending with the last byte of the last sector of that track. The sectors containing the volume recognition space shall have the following attributes (see 3/A.1):

– all sectors shall have the Mode 1 attribute, or the first 16 sectors of the track shall have the Mode 2 Form 1 attribute;

– all the sectors shall be data sectors.

NOTE

These restrictions ensure that the volume recognition area (see 2/8.3) begins in the seventeenth sector of a data track.

The first 16 sectors of the volume recognition space shall be designated as the reserved session area. Part 3 does not specify the interpretation of any information recorded in the reserved session area.

9.1.7 Volume recognition sequence

At least one CD-WO Extended Area shall be recorded in each volume recognition sequence (see 2/8.3.1) of a volume according to the schema shown in figure 3/1.

Within a volume recognition sequence, the CD-WO Extended Area that begins in the sector with the highest sector number shall be referred to as the prevailing CD-WO Extended Area.

```
[CD-WO Extended Area]{
      <Beginning Extended Area Descriptor> 1+
      <Volume Descriptor Set>
      <File System Descriptor Set> 0+1
      <Terminating Descriptor> 1+
}
```

Figure 1 - CD-WO Extended Area schema

NOTE

Part 3 allows the creation of media which conform to both this Part and ECMA-119.

9.2 Volume set

A volume set shall consist of one or more volumes having a volume set identification common to all volumes of the volume set. The volumes of 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. The largest volume sequence number in the volume set shall be referred to as the assigned volume set size.

Each Primary Volume Descriptor (see 3/10.1) recorded on a volume specifies a volume set identification consisting of the contents of the Volume Set Identifier and Descriptor Character Set fields and a volume identification consisting of the contents of the Volume Identifier and Descriptor Character Set fields. The Primary Volume Descriptor in the prevailing Volume Descriptor Set (see 3/9.4.2) of each volume of a volume set shall specify the volume identification and the same volume set identification. The same volume identification shall not be specified by more than one volume of a volume set.

NOTE

ECMA-119 uses the term "volume group" to refer to volumes within a volume set that had their contents "established at the same time" and are consecutively numbered in the Volume Sequence Number field of the Primary Volume Descriptor. The volume sequence number of the volume that has the highest volume sequence number in the volume group is recorded as the assigned volume set size in the Volume Set Size field of the Primary Volume Descriptor for each volume of the volume group.

In ECMA-119, each volume of a volume group contains a description of all the directories and files that are recorded on those volumes of the volume set having a volume sequence number less than, or equal to, the assigned volume set size of the volume group.

It is important to note that the concept of a volume group does not exist in this Part.

9.3 Volume structure descriptors

Characteristics of a volume and a volume set shall be specified by volume structure descriptors. A volume structure descriptor shall be one of the following types: (see 3/8.7)

- Primary Volume Descriptor (see 3/10.1)
- Supplementary Volume Descriptor (see 3/10.2)
- Volume Partition Descriptor (see 3/10.3)
- End Transaction Descriptor (see 3/10.4)
- Terminating Descriptor (see 3/10.5)
- File Set Descriptor (see 3/12.1)
- Implementation Use Descriptor (see 3/12.2).

Volume structure descriptors shall be recorded as specified in 3/9.4, 3/9.6 and 3/11.2.

9.3.1 Recording of volume structure descriptors

Each volume structure descriptor shall be recorded starting at the first byte of a sector. The location, or address, of a volume structure descriptor shall be the sector number of the sector containing the first byte of the descriptor. All space, if any, after the end of a volume structure descriptor up to the end of the sector is reserved for future standardisation and shall be recorded as all #00 bytes.

9.4 Volume Descriptor Set

A Volume Descriptor Set shall be recorded according to the schema shown in figure 3/2.

```
[Volume Descriptor Set]{
      <Primary Volume Descriptor>1+
      {
            <Supplementary Volume Descriptor> |
            <Volume Partition Descriptor> |
            <End Transaction Descriptor>
      }0+
      <Terminating Descriptor>1+
}
```

Figure 2 - Volume Descriptor Set schema

A Volume Descriptor Set shall begin with a sequence of one or more identical Primary Volume Descriptors. A Primary Volume Descriptor shall identify the volume, the volume set to which the volume belongs, the sequence number of the volume within the volume set, attributes of the volume, the character set used in recording the contents of certain fields within the Primary Volume Descriptor, and the rule for recording and locating the prevailing End Transaction Descriptor (see 3/9.6.3).

A Volume Descriptor Set may contain zero or more Supplementary Volume Descriptors, each recorded at least once. A Supplementary Volume Descriptor shall provide an alternate identification of the volume and the volume set to which it belongs. Each Supplementary Volume Descriptor shall have an assigned Supplementary Volume Descriptor Sequence Number (see 3/10.2.8). The set of all Supplementary Volume Descriptor Sequence Numbers in a Volume Descriptor Set shall form a continuous ascending sequence of integers starting from 1. All Supplementary Volume Descriptors with identical Supplementary Volume Descriptor Sequence Numbers shall have identical contents. Supplementary Volume Descriptors with different Supplementary Volume Descriptor Sequence Numbers shall not have the same contents in the Volume Identifier (see 3/10.2.6) and Descriptor Character Set (see 3/10.2.5) fields.

A Volume Descriptor Set may contain zero or more Volume Partition Descriptors, each recorded at least once. A Volume Partition Descriptor shall specify a partition (see 3/9.1.4.3) of the volume, and the attributes and identification of the volume partition. All the Volume Partition Descriptors with identical contents for their Descriptor Character Set (see 3/10.3.5) and Volume Partition Identifier (see 3/10.3.7) fields shall have identical contents.

A Volume Descriptor Set may contain zero or more End Transaction Descriptors. All the End Transaction Descriptors, except for their End Transaction Descriptor Location (see 3/10.4.6) fields, shall have identical contents.

The sequence shall be terminated by a sequence of one or more identical Terminating Descriptors (see 3/10.5).

9.4.1 Recording of Volume Descriptor Sets

The descriptors of a Volume Descriptor Set shall be recorded in a sequence of sectors having consecutive ascending addresses. The location of a Volume Descriptor Set shall be the location of the first descriptor in the Volume Descriptor Set. A Volume Descriptor Set shall end in the track in which it begins.

9.4.2 Prevailing Volume Descriptor Set

The Volume Descriptor Set contained in the prevailing CD-WO Extended Area (see 3/9.1.7) in the volume recognition sequence of a volume shall be referred to as the prevailing Volume Descriptor Set of the volume.

The information in the prevailing Volume Descriptor Set shall override the information in all other Volume Descriptor Sets of the volume on which the prevailing Volume Descriptor Set is recorded.

9.5 Volume space management

9.5.1 Volume Space Table

For each volume in a volume set, there shall be a Volume Space Table which shall specify the size and attribute information for each track in the volume and, optionally, a description of the unallocated sectors in the volume.

The Volume Space Tables Location Directory and all the Volume Space Tables identified by the Volume Space Tables Location Directory shall be recorded on the volume on which the prevailing End Transaction Descriptor is recorded.

9.5.2 Recording of Volume Space Tables

A Volume Space Table shall be recorded as a contiguous sequence of Track Specification Records (see 3/10.6) in a file. The Track Specification Records in a Volume Space Table shall be recorded in ascending order of the value of the Track Number field (see 3/10.6.2).

Each Track Specification Record in a Volume Space Table shall have a different track number in the Track Number field.

9.5.3 Identification of Volume Space Tables

Volume Space Tables shall be identified by the prevailing End Transaction Descriptor of the volume set (see 3/9.6.3). There shall be one Volume Space Table identified for each volume of the volume set having an assigned volume sequence number less than or equal to that of the volume on which the Volume Space Tables are recorded. Each Volume Space Table shall be recorded as the contents of a separate file as specified in 3/9.5.2.

A Directory Record (see 3/15.1) shall identify each such file as follows:

– the File Identifier field of the Directory Record shall contain the assigned volume sequence number of the volume to which the Volume Space Table applies. This field shall be a 4-byte field recorded as a `Uint16BOTH` (see 3/8.3).

– the Length of File Identifier field of the Directory Record shall be set to 4.

– the File Version Number field of the Directory Record shall be set to 1.

These files shall not be identified by any directory (see 3/13) of any file set (see 3/11.1) recorded in the volume set.

9.5.4 Volume Space Tables Location Directory

The Volume Space Tables Location Directory shall be recorded as a directory (see 3/13.1). The directory shall identify, as specified in 3/9.5.3, a file for each Volume Space Table.

A Path Table Record (see 3/15.2) shall identify the directory as follows:

– the File Identifier field of the Path Table Record shall contain 0. This field shall be a 4-byte field recorded as a `Uint16BOTH` (see 3/8.3).

– the Length of File Identifier field of the Path Table Record shall be set to 4.

– the Parent Directory Number field of the Path Table Record shall be set to 0.

The Volume Space Tables Location Directory need not be recorded if there is only one volume in the volume set.

The Volume Space Tables Location Directory shall not be identified by any path table (see 3/13.3) of any file set (see 3/11.1) recorded in the volume set.

9.6 Transactions

The information on a volume set shall be recorded as a sequence of transactions. A transaction shall be recorded over one or more sectors of the volume set. The data sectors of the transaction need not have consecutive sector numbers and may be recorded on one or more volumes of the volume set.

The order of the recording of the data sectors in which the information of a transaction is contained shall be subject to the restrictions, if any, of the standard for recording.

9.6.1 Identification of a transaction

A transaction shall be identified by an End Transaction Descriptor. Each End Transaction Descriptor shall have an assigned End Transaction Descriptor transaction number. The set of all End Transaction Descriptor transaction numbers in a volume shall form a continuous ascending sequence of integers starting from 1. End Transaction Descriptors with identical End Transaction Descriptor transaction numbers in a volume shall have identical contents, except for the values of their End Transaction Descriptor Location fields (see 3/10.4.6).

9.6.2 End Transaction Descriptor

An End Transaction Descriptor shall identify:

- the assigned transaction number of the End Transaction Descriptor.
- the location of the prevailing Volume Descriptor Set of the volume at the time of recording the End Transaction Descriptor.
- the Volume Space Table recorded on the volume for each volume of the volume set having an assigned volume sequence number less than or equal to that of the volume on which the End Transaction Descriptor is recorded.
- the path tables (see 3/13.3) recorded on the volume for the file sets identified by all of the File Set Descriptors (see 3/12.1) identified by the prevailing File System Descriptor Set (see 3/11.2.2).
- the location of the prevailing File System Descriptor Set of the volume set at the time of recording the End Transaction Descriptor.
- the location of the End Transaction Descriptor, if any, of the volume whose transaction number is one less than the assigned transaction number of this End Transaction Descriptor.
- an indication that a volume has been added to the volume set of which the volume that the End Transaction Descriptor is recorded on is a member (see 3/10.4.12).

An End Transaction Descriptor shall be recorded one or more times after the recording of each transaction to identify the transaction. The End Transaction Descriptors so recorded shall have identical contents except for the values of their End Transaction Descriptor Location fields.

9.6.3 Prevailing End Transaction Descriptor

The End Transaction Descriptor with the greatest transaction number recorded on the volume having the greatest assigned volume sequence number of a volume set shall be the prevailing End Transaction Descriptor of the volume set. The prevailing End Transaction Descriptor of a volume set shall be the last recorded data sector of the track in which the prevailing End Transaction Descriptor is recorded.

9.6.4 End Transaction Track

A track of the volume having the greatest assigned volume sequence number of a volume set shall be the assigned End Transaction Track (see 3/10.6.5). The assigned End Transaction Track shall be the track in which the latest End Transaction Descriptor of the volume set is recorded and the track in which next End Transaction Descriptor shall be recorded.

NOTE

Descriptors other than End Transaction Descriptors and data sectors of a transaction may be recorded on the End Transaction Track.

9.6.5 End Transaction Descriptor recording rule

The Primary Volume Descriptor recorded in the prevailing Volume Descriptor Set of the volume having the greatest volume sequence number of the volume set shall specify the rule (see 3/10.1.15) for recording the prevailing End Transaction Descriptor of the volume set.

The sector on which the prevailing End Transaction Descriptor of a volume set is recorded shall always be the last recorded data sector of the End Transaction Track.

NOTE

The purpose of the End Transaction Descriptor recording rule is to expedite the process of locating the prevailing End Transaction Descriptor of a volume set by specifying where the prevailing End Transaction Descriptor shall be recorded. Some End Transaction Descriptor recording rules may allow finding the prevailing End Transaction Descriptor of a volume set quickly without having to read the last recorded data sector of every data track in the volume having the greatest volume sequence number of the volume set.

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 table 3/3.

Table 3 - Primary Volume Descriptor format

BP	Length	Name	Contents
0	1	Structure Type	`Uint8` (1/7.1.1) = 1
1	5	Standard Identifier	bytes = "CDW02"
6	1	Structure Version	`Uint8` (1/7.1.1) = 2
7	1	Reserved	`#00 byte`
8	64	Descriptor Character Set	`charspec` (1/7.2.1)
72	32	Implementation Identifier	`regid` (1/7.4)
104	32	Volume Identifier	`dstring` (1/7.2.12)
136	128	Volume Set Identifier	`dstring` (1/7.2.12)
264	4	Volume Set Size	`Uint16BOTH` (3/8.3)
268	4	Volume Sequence Number	`Uint16BOTH` (3/8.3)
272	8	Logical Block Size	`Uint32BOTH` (3/8.6)
280	4	Control Flags	`Uint32` (1/7.1.5)
284	4	End Transaction Track	`Uint16BOTH` (3/8.3)
288	8	Prevailing End Transaction Descriptor Location	`Uint32BOTH` (3/8.6)
296	8	End Transaction Descriptor Recording Rule	`Uint32BOTH` (3/8.6)
304	4	Maximum Interchange Level	`Uint16BOTH` (3/8.3)
308	4	Maximum Character Set List	`Uint32` (1/7.1.5)
312	12	Volume Set Creation Date and Time	`timestamp` (1/7.3.1)
324	12	Descriptor Recording Date and Time	`timestamp` (1/7.3.1)
336	512	Application Use	bytes
848	1 200	Reserved	#00 bytes

10.1.1 Structure Type (BP 0)

This field shall specify 1.

10.1.2 Standard Identifier (BP 1)

This field shall specify "CDW02".

10.1.3 Structure Version (BP 6)

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

10.1.4 Reserved (BP 7)

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

10.1.5 Descriptor Character Set (BP 8)

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

10.1.6 Implementation Identifier (BP 72)

This field shall specify an identification of an implementation which can recognise and act upon the contents of the reserved session area (see 3/9.1.6) of the volume. If this field contains all #00 bytes, then no such implementation is identified.

10.1.7 Volume Identifier (BP 104)

This field shall specify an identification of the volume.

10.1.8 Volume Set Identifier (BP 136)

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

10.1.9 Volume Set Size (BP 264)

This field shall specify the assigned volume set size (see 3/9.2) of the volume when this descriptor is recorded.

10.1.10 Volume Sequence Number (BP 268)

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

10.1.11 Logical Block Size (BP 272)

This field shall specify the size, in bytes, of a logical block.

10.1.12 Control Flags (BP 280)

This field shall specify the control attributes of the Primary Volume Descriptor as shown in table 3/4.

Table 4 - Control Flags interpretation

Bit	Interpretation
0	If set to ONE, shall mean that each Volume Descriptor Set of the volume recorded after the recording of this Primary Volume Descriptor shall contain a Primary Volume Descriptor with contents identical to this Primary Volume Descriptor, except that the contents of the End Transaction Track, Prevailing End Transaction Descriptor Location and End Transaction Descriptor Recording Rule fields shall be recorded according to clauses 3/10.1.13, 3/10.1.14 and 3/10.1.15. If set to ZERO, shall mean that no such restriction is specified.
1-31	Reserved for future standardisation and shall be set to ZERO.

10.1.13 End Transaction Track (BP 284)

This field shall specify the track number of the assigned End Transaction Track (see 3/9.6.4) of this volume when this descriptor is recorded. A value of 0 shall mean no such track number is specified.

This field shall be non-zero if this descriptor is recorded in the prevailing Volume Descriptor Set on the volume having the greatest volume sequence number of the volume set, and if the value of the End Transaction Recording Rule field (see 3/10.1.15) is either 1 or 2.

10.1.14 Prevailing End Transaction Descriptor Location (BP 288)

This field shall specify the sector number of the sector in which the prevailing End Transaction Descriptor of the volume set is recorded when this Primary Volume Descriptor is recorded if the volume on which the Primary Volume Descriptor is recorded is the highest numbered volume of the volume set. A value of 0 shall mean no such sector number is specified.

This field shall be non-zero if this descriptor is recorded in the prevailing Volume Descriptor Set on the volume having the greatest volume sequence number of the volume set, and if the End Transaction Recording Rule field is 1. The sector number specified in this field shall be in the track specified in the End Transaction Track field (see 3/10.1.13) of this descriptor.

10.1.15 End Transaction Descriptor Recording Rule (BP 296)

If this descriptor is recorded in the prevailing Volume Descriptor Set on the volume having the greatest volume sequence number of the volume set, this field shall specify the recording rule to record the prevailing End Transaction Descriptor of the volume set (see 3/9.6.3). Otherwise, the information recorded in this field shall be ignored.

The End Transaction Descriptor recording rule shall be specified as shown in table 3/5.

Table 5 - End Transaction Descriptor Recording Rule interpretation

Rule	Interpretation
0	The location of the prevailing End Transaction Descriptor is not specified by this field. *NOTE* *This rule requires reading the last recorded data sector in every data track on the volume on which the Primary Volume Descriptor is recorded to find the prevailing End Transaction Descriptor of the volume set*
1	The value in the prevailing End Transaction Descriptor Location field (3/10.1.14) in this descriptor shall be non-zero and shall identify, for the volume on which the Primary Volume Descriptor is recorded, the sector number of the prevailing End Transaction Descriptor of the volume set. *NOTE* *This rule is intended to be used for read-only media. For performance reasons, the prevailing End Transaction Descriptor identified in the Prevailing End Transaction Descriptor Location field may be recorded in the same Volume Descriptor Set with this Primary Volume Descriptor, provided there is an End Transaction Descriptor recorded on the last recorded data sector of the End Transaction Track of the volume.*
2	The prevailing End Transaction Descriptor shall be recorded in the track identified by the End Transaction Track field of this Primary Volume Descriptor. The End Transaction Track of the volume may be reassigned subject to the following rule: If the End Transaction Track (track number *x*) is reassigned to another track (track number *y*), the End Transaction Descriptor in the last recorded data sector of track number *x* shall specify the track number *y* in the End Transaction Descriptor Track field (3/10.4.11), and the prevailing End Transaction Descriptor shall be recorded in track number *y*. No further recording may occur in track number *x* unless track number *x* is again assigned to be the End Transaction Track.
3	The prevailing End Transaction Descriptor shall be recorded, on the volume on which the Primary Volume Descriptor is recorded, in a sector number that is higher than the sector number of all previously recorded End Transaction Descriptors of the volume. *NOTE* *This rule is designed for track-at-once recording using the backward searching strategy.*
4-2 047	Reserved for future standardisation.
2 048 and above	Reserved for implementation use.

10.1.16 Maximum Interchange Level (BP 304)

This field shall specify the maximum value that may be specified for the Maximum Interchange Level field of any File Set Descriptor (see 3/12.1) in the volume set.

10.1.17 Maximum Character Set List (BP 308)

This field shall identify the character sets (see 1/7.2.11) that may be specified by any field, whose contents are specified to be a `charspec` (see 1/7.2.1), of any descriptor specified in this Part and recorded on the volume set identified by this Primary Volume Descriptor.

NOTE

The Maximum Interchange Level 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.

10.1.18 Volume Set Creation Date and Time (BP 312)

This field shall specify the date and time of the day of recording the first Primary Volume Descriptor recorded on the volume having the volume sequence number 1 in the volume set.

10.1.19 Descriptor Recording Date and Time (BP 324)

This field shall specify the date and time of the day when this descriptor is recorded.

NOTE

If the recording of this descriptor is under the restrictions specified by bit 0 of the Control Flags field, this field shall contain the same value as recorded in the Descriptor Recording Date and Time field of the first Primary Volume Descriptor that set bit 0 of the Control Flags field to ONE.

10.1.20 Application Use (BP 336)

This field shall be used to store application use extended attributes (see 3/14). Application use extended attributes shall be recorded contiguously according to the extended attribute format (see 3/14.2). Any remaining bytes in this field shall be set to #00.

If this field contains all #00 bytes., then no such extended attributes are recorded.

NOTE

The “CE” (see 3/15.3.9) extended attribute may be recorded in the Application Use field to provide an additional extended attribute area to store more application use extended attributes.

10.1.21 Reserved (BP 848)

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

10.2 Supplementary Volume Descriptor

The Supplementary Volume Descriptor shall specify an alternate identification of the volume and the volume set to which it belongs. It shall be recorded in the format shown in table 3/6.

Table 6 - Supplementary Volume Descriptor format

BP	Length	Name	Contents
0	1	Structure Type	`Uint8` (1/7.1.1) = 2
1	5	Standard Identifier	bytes = “CDW02”
6	1	Structure Version	`Uint8` (1/7.1.1) = 2
7	1	Reserved	#00 byte
8	64	Descriptor Character Set	`charspec` (1/7.2.1)
72	32	Volume Identifier	`dstring` (1/7.2.12)
104	128	Volume Set Identifier	`dstring` (1/7.2.12)
232	4	Supplementary Volume Descriptor Sequence Number	`Uint16BOTH` (3/8.3)
236	4	Control Flags	`Uint32` (1/7.1.5)
240	12	Descriptor Recording Date and Time	`timestamp` (1/7.3.1)
252	512	Application Use	bytes
764	1 284	Reserved	#00 bytes

10.2.1 Structure Type (BP 0)

The number in this field shall specify 2.

10.2.2 Standard Identifier (BP 1)

This field shall specify “CDW02”.

10.2.3 Structure Version (BP 6)

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

10.2.4 Reserved (BP 7)

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

10.2.5 Descriptor Character Set (BP 8)

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

10.2.6 Volume Identifier (BP 72)

This field shall specify an identification of the volume.

10.2.7 Volume Set Identifier (BP 104)

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

10.2.8 Supplementary Volume Descriptor Sequence Number (BP 232)

This field shall specify the assigned Supplementary Volume Descriptor Sequence Number (see 3/9.4) of this descriptor.

10.2.9 Control Flags (BP 236)

This field shall specify the control attributes of the Supplementary Volume Descriptor as shown in table 3/7.

Table 7 - Control Flags interpretation

Bit	Interpretation
0	If set to ONE, shall mean that each Volume Descriptor Set of the volume recorded after the recording of this Supplementary Volume Descriptor shall contain a Supplementary Volume Descriptor with contents identical to this Supplementary Volume Descriptor. If set to ZERO, shall mean that no such restriction is specified.
1-31	Reserved for future standardisation and shall be set to ZERO.

10.2.10 Descriptor Recording Date and Time (BP 240)

This field shall specify the date and time of the day when this descriptor is recorded.

NOTE

If the recording of this descriptor is under the restrictions specified by bit 0 of the Control Flags field, this field shall contain the same value as recorded in the Descriptor Recording Date and Time field of the first Supplementary Volume Descriptor that set bit 0 of the Control Flags field to ONE.

10.2.11 Application Use (BP 252)

This field shall be used to store application use extended attributes (see 3/14). Application use extended attributes shall be recorded contiguously according to the extended attribute format (see 3/14.2). Any remaining bytes in this field shall be set to #00.

If this field contains all #00 bytes., then no such extended attributes are recorded.

NOTE

The "CE" extended attribute may be recorded in the Application Use field to provide an additional extended attribute area to store more application use extended attributes.

10.2.12 Reserved (BP 764)

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

10.3 Volume Partition Descriptor

The Volume Partition Descriptor shall identify a volume partition within the volume space, an implementation which can recognise and act upon the contents of fields in the descriptor reserved for implementation use, the position and the size of the volume partition, and the version of the standard which applies to the Volume Partition Descriptor.

The Volume Partition Descriptor shall be recorded in the format shown in table 3/8.

Table 8 - Volume Partition Descriptor format

BP	Length	Name	Contents
0	1	Structure Type	`Uint8` (1/7.1.1) = 5
1	5	Standard Identifier	bytes = “CDW02”
6	1	Structure Version	`Uint8` (1/7.1.1) = 2
7	1	Reserved	#00 byte
8	64	Descriptor Character Set	`charspec` (1/7.2.1)
72	32	Implementation Identifier	`regid` (1/7.4)
104	32	Volume Partition Identifier	`dstring` (1/7.2.12)
136	8	Volume Partition Location	`Uint32BOTH` (3/8.6)
144	8	Volume Partition Size	`Uint32BOTH` (3/8.6)
152	1 896	Implementation Use	bytes

10.3.1 Structure Type (BP 0)

The number in this field shall specify 5.

10.3.2 Standard Identifier (BP 1)

This field shall specify “CDW02”.

10.3.3 Structure Version (BP 6)

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

10.3.4 Reserved (BP 7)

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

10.3.5 Descriptor Character Set (BP 8)

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

10.3.6 Implementation Identifier (BP 72)

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.

10.3.7 Volume Partition Identifier (BP 104)

This field shall specify an identification of the volume partition.

10.3.8 Volume Partition Location (BP 136)

This field shall specify the logical block number of the first logical block allocated to the volume partition.

10.3.9 Volume Partition Size (BP 144)

This field shall specify the number of logical blocks in the volume partition.

10.3.10 Implementation Use (BP 152)

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

10.4 End Transaction Descriptor

The End Transaction Descriptor shall identify a transaction and certain attributes of that transaction. It shall be recorded in the format shown in table 3/9.

Table 9 - End Transaction Descriptor format

BP	Length	Name	Contents
0	1	Structure Type	`Uint8` (1/7.1.1) = 6
1	5	Standard Identifier	bytes = “CDW02”
6	1	Structure Version	`Uint8` (1/7.1.1) = 2
7	1	End Transaction Flags	`Uint8` (1/7.1.1)
8	64	Reserved	#00 bytes
72	8	End Transaction Descriptor Location	`Uint32BOTH` (3/8.6)
80	8	Prevailing Volume Descriptor Set Location	`Uint32BOTH` (3/8.6)
88	8	Prevailing File System Descriptor Set Location	`Uint32BOTH` (3/8.6)
96	8	Previous Prevailing File System Descriptor Set Location	`Uint32BOTH` (3/8.6)
104	8	Previous End Transaction Descriptor Location	`Uint32BOTH` (3/8.6)
112	4	End Transaction Track	`Uint16BOTH` (3/8.3)
116	4	Last Volume of Volume Set	`Uint16BOTH` (3/8.3)
120	8	Transaction Number	`Uint32BOTH` (3/8.6)
128	12	Descriptor Recording Date and Time	`timestamp` (1/7.3.1)
140	4	Number of File Set Descriptors	`Uint16BOTH` (3/8.3)
144	256	Volume Space Tables Information	bytes
400	256	Path Tables Information	bytes
656	1 392	Reserved	#00 bytes

10.4.1 Structure Type (BP 0)

The number in this field shall specify 6.

10.4.2 Standard Identifier (BP 1)

This field shall specify “CDW02”.

10.4.3 Structure Version (BP 6)

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

10.4.4 End Transaction Flags (BP 7)

This field shall specify certain characteristics of this descriptor as shown in table 3/10.

Table 10 - End Transaction Flags interpretation

Bit	Interpretation
0	If set to ZERO, shall mean this volume may have the greatest volume sequence number of the volume set. If set to ONE, shall mean that this volume shall not have the greatest volume sequence number of the volume set. *NOTE* *This identifies whether the volume is definitely not the last volume of the volume set at the time the End Transaction Descriptor is recorded.*
1	If set to ZERO, shall mean the End Transaction Descriptor identifies a transaction (see 3/9.6) for which all information of the transaction has been recorded successfully on one or more volumes of the volume set having an assigned volume sequence number less than, or equal to, the assigned volume sequence number of the volume on which the End Transaction Descriptor is recorded. If set to ONE, shall mean the End Transaction Descriptor identifies a transaction for which some of the information of the transaction is to be recorded on one or more volumes with higher volume sequence numbers than the volume on which the End Transaction Descriptor is recorded. If this bit is set to ONE, bit 0 of this field shall also be set to ONE. *NOTE* *The setting of this bit indicates the recording of a transaction was not completed on this volume, most likely caused by insufficient space on the volume. The completion of the recording of the transaction requires adding additional volumes to the volume set. The remaining information of the incomplete transaction is to be recorded and identified by another End Transaction Descriptor on a volume with a higher volume sequence number. If bit 1 of this field is set to ONE, bit 0 of this field shall also be set to ONE.*
2	If set to ZERO, the track number in the End Transaction Track field (3/10.4.11) shall be the same as the track number of the track in which this End Transaction Descriptor is recorded. If set to ONE, shall mean that there shall be an End Transaction Descriptor with a higher End Transaction Descriptor transaction number recorded on a different track, the track number of which shall be specified by the End Transaction Track field. This bit shall be set to ZERO if the value of the End Transaction Recording Rule field in the Primary Volume Descriptor recorded in the prevailing Volume Descriptor Set is not set to 2 (3/10.1.15).
3-7	Reserved for future standardisation and shall be set to ZERO.

10.4.5 Reserved (BP 8)

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

10.4.6 End Transaction Descriptor Location (BP 72)

This field shall specify the sector number of the location of this descriptor.

10.4.7 Prevailing Volume Descriptor Set Location (BP 80)

This field shall specify the sector number of the location of the prevailing Volume Descriptor Set of the volume when this descriptor is recorded.

10.4.8 Prevailing File System Descriptor Set Location (BP 88)

This field shall specify the sector number of the location of the prevailing File System Descriptor Set of the volume when this descriptor is recorded.

10.4.9 Previous Prevailing File System Descriptor Set Location (BP 96)

This field shall specify the sector number recorded in the Prevailing File System Descriptor Set Location field of the End Transaction Descriptor whose transaction number is 1 less than the transaction number of this descriptor. This field shall be set to 0 if no such descriptor is recorded.

10.4.10 Previous End Transaction Descriptor Location (BP 104)

This field shall specify the sector number recorded in the End Transaction Descriptor Location field of the End Transaction Descriptor whose transaction number is 1 less than the transaction number of this descriptor. This field shall be set to 0 if no such descriptor is recorded.

10.4.11 End Transaction Track (BP 112)

If bit 2 of the End Transaction Flags (see 3/10.4.4) is ZERO, this field shall specify the track number of the End Transaction Track (see 3/9.6.4) of the volume when the End Transaction Descriptor is recorded.

If bit 2 of the End Transaction Flags is ONE, this field shall specify the track number of the newly assigned End Transaction Track of the volume when the End Transaction Descriptor is recorded. The track number of the newly assigned End Transaction Track shall not be the same as the track number of the track in which this End Transaction Descriptor is recorded.

10.4.12 Last Volume of Volume Set (BP 116)

This field shall specify the volume sequence number (see 3/9.2) of the volume having the greatest sequence number of the volume set when this descriptor is recorded.

NOTE

If the number in this field is the same as the volume sequence number of the volume on which the End Transaction Descriptor is recorded, that does not mean that there are no further volumes in the volume set. It may not have been possible to add an End Transaction Descriptor to the volume to indicate the addition of more volumes to the volume set as there may have been insufficient space to do so or the media might be read-only.

10.4.13 Transaction Number (BP 120)

This field shall specify the transaction number (see 3/9.6.1) of the transaction identified by this descriptor.

10.4.14 Descriptor Recording Date and Time (BP 128)

This field shall specify the date and time of the day when this descriptor is recorded.

NOTE

This field can be used to indicate the last modification date and time of this volume set.

10.4.15 Number of File Set Descriptors (BP 140)

This field shall specify the greatest assigned File Set Descriptor Sequence Number of the volume set when this descriptor is recorded.

10.4.16 Volume Space Tables Information (BP 144)

This field shall identify the Volume Space Tables of the volume set as follows starting at the first byte of this field:

If there is only one volume in the volume set, this field shall contain either a Directory Record (see 3/9.5.3) describing the Volume Space Table or a Path Table Record (see 3/9.5.4) describing the Volume Space Tables Location Directory.

If there is more than one volume in the volume set, this field shall contain a Path Table Record (see 3/9.5.4) describing the Volume Space Tables Location Directory.

NOTE

Byte 25 of this field is the location of the File Flags field in both the Path Table and Directory Records. Bit 1 of the File Flags field is the Directory bit. The Directory bit of the File Flags field in a Path Table Record shall always be set to ONE (see 3/15.2.5).

Unused bytes in this field not occupied by the Directory Record or Path Table Record shall be set to #00.

10.4.17 Path Tables Information (BP 400)

This field shall identify the path tables of the file set identified by the prevailing File System Descriptor Set (see 3/11.2.2) as follows starting at the first byte of this field:

If there is only one file set identified by the prevailing File System Descriptor Set of the volume set, this field shall contain either a Directory Record (see 3/13.3.3) describing the path table or a Path Table Record (see 3/13.3.4) describing the Path Tables Location Directory.

If there is more than one file set identified by the prevailing File System Descriptor Set of the volume set, this field shall contain a Path Table Record (see 3/13.3.4) describing the Path Tables Location Directory.

NOTE

Byte 25 of this field is the location of the File Flags field in both the Path Table and Directory Records. Bit 1 of the File Flags field is the Directory bit. The Directory bit of the File Flags field in a Path Table Record shall always be set to ONE (see 3/15.2.5).

Unused bytes in this field not occupied by the Directory Record or Path Table Record shall be set to #00.

10.4.18 Reserved (BP 656)

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

10.5 Terminating Descriptor

The Terminating Descriptor shall be recorded in the format shown in table 3/11.

Table 11 - Terminating Descriptor format

BP	Length	Name	Contents
0	1	Structure Type	`Uint8` (1/7.1.1) = 255
1	5	Standard Identifier	bytes = “CDW02”
6	1	Structure Version	`Uint8` (1/7.1.1) = 2
7	1	Control Flags	`Uint8` (1/7.1.1)
8	2 040	Reserved	#00 bytes

10.5.1 Structure Type (BP 0)

The number in this field shall specify 255.

10.5.2 Standard Identifier (BP 1)

This field shall specify “CDW02”.

10.5.3 Structure Version (BP 6)

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

10.5.4 Control Flags (BP 7)

This field shall specify the control attributes of the Volume Descriptor Set as shown in table 3/12.

Table 12 - Control Flags interpretation

Bit	Interpretation
0	If set to ONE and if this Terminating Descriptor is recorded in a Volume Descriptor Set, shall mean that no new Supplementary Volume Descriptors or Volume Partition Descriptors shall be added to any Volume Descriptor Set of the volume recorded after the recording of this Terminating Descriptor. If set to ONE and if this Terminating Descriptor is recorded in a File System Descriptor Set, shall mean that no new File Set Descriptors or Implementation Use Descriptors shall be added to any File System Descriptor Set of the volume set recorded after the recording of this Terminating Descriptor. If set to ZERO, shall mean that no such restriction is specified.
1-7	Reserved for future standardisation and shall be set to ZERO.

10.5.5 Reserved (BP 8)

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

10.6 Track Specification Record

The Track Specification Record shall identify a track and the attributes of the track. It shall be recorded in the format shown in table 3/13.

Table 13 - Track Specification Record format

RBP	Length	Name	Contents
0	4	Session Number	`Uint16BOTH` (3/8.3)
4	4	Track Number	`Uint16BOTH` (3/8.3)
8	2	Track Type	`Uint16` (1/7.1.3)
10	1	Track Contents	`Uint8` (1/7.1.1)
11	1	Track Flags	`Uint8` (1/7.1.1)
12	8	Packet Size	`Uint32BOTH` (3/8.6)
20	8	Start Location of Track	`Uint32BOTH` (3/8.6)
28	8	End Location of Track	`Uint32BOTH` (3/8.6)
36	8	Last Written Sector	`Uint32BOTH` (3/8.6)

10.6.1 Session Number (RBP 0)

This field shall specify the assigned session number (see 3/9.1.3) of the session in which the track is recorded.

If the value of the Track Contents field is 255, this field shall be set to 0.

10.6.2 Track Number (RBP 4)

This field shall specify the assigned track number (see 3/9.1.2.1) of the track.

If the value of the Track Contents field is 255, this field shall be set to 0.

10.6.3 Track Type (RBP 8)

This field shall specify the attributes of the track as shown in table 3/14.

If the value of the Track Contents field is not 1, this field shall be set to 0.

NOTE

Table 3/15 summarises the possible sector attributes of a track. Table 3/16 summarises the possible recording methods for a track.

Table 14 - Track Type interpretation

Bit	Interpretation
0	If set to ZERO, the track shall not contain any sector with the Mode 0 attribute (3/A.1). If set to ONE, the track shall contain only sectors with the Mode 0 attribute (3/A.1).
1	If set to ZERO, the track shall not contain any sector with the Mode 1 attribute (3/A.1). If set to ONE, the track shall contain only sectors with the Mode 1 attribute (3/A.1).
2	If set to ZERO, the track shall not contain any sector with the Mode 2 attribute (3/A.1). If set to ONE, the track shall contain only sectors with the Mode 2 attribute (3/A.1). If either bit 0 or bit 1 is set to ONE, bit 2 shall be set to ZERO.
3	If set to ZERO, the track shall not contain any sector with the Mode 2 Form 1 attribute (3/A.1). If set to ONE, the track may contain sectors with the Mode 2 Form 1 attribute (3/A.1) and bit 2 of this field shall be set to ONE. If either bit 0 or bit 1 is set to ONE, bit 3 shall be set to ZERO.
4	If set to ZERO, the track shall not contain any sector with the Mode 2 Form 2 attribute (3/A.1). If set to ONE, the track may contain sectors with the Mode 2 Form 2 attribute (3/A.1) and bit 2 of this field shall be set to ONE. If either bit 0 or bit 1 is set to ONE, bit 4 shall be set to ZERO.
5-8	These four bits shall be interpreted as a 4 bit binary number as follows: 0 shall mean that the track is read-only and not available for recording. 1 shall mean that the track is write-once and may be available for recording. 2-15 are reserved for future standardisation.
9	If set to ZERO, the track shall be recorded in track-at-once mode and bits 10 and 11 shall be ZERO. If set to ONE, the track shall be recorded in incremental mode and exactly one of bits 10 and 11 shall be ONE.
10	If set to ZERO, the track shall not be recorded with fixed-length packets. If set to ONE, the track shall be recorded with fixed-length packets, and bit 9 shall be set to ONE and bit 11 shall be set to ZERO.
11	If set to ZERO, the track shall not be recorded with variable-length packets. If set to ONE, the track shall be recorded with variable-length packets, and bit 9 shall be set to ONE and bit 10 shall be set to ZERO.
12-15	Reserved for future standardisation and shall be set to ZERO.

Table 15 - Possible sector attributes in a track

BIT:	4	3	2	1	0	Sector Attributes in a Track
	0	0	0	0	1	only Mode 0
	0	0	0	1	0	only Mode 1
	0	0	1	0	0	only Mode 2
	0	1	1	0	0	only Mode 2 Form 1
	1	0	1	0	0	only Mode 2 Form 2
	1	1	1	0	0	Mode 2 Form 1, Mode 2 Form 2

Table 16 - Possible recording methods for a track

BIT: 11	10	9	Recording Method
0	0	0	track-at-once
0	1	1	incremental with fixed-length packets
1	0	1	incremental with variable-length packets

10.6.4 Track Contents (RBP 10)

This field shall specify the contents of the track as shown in table 3/17.

Table 17 - Track Contents Interpretation

Contents	Interpretation
0	The track contents are not specified by this field.
1	The Track Type field shall specify the characteristics of the track.
2	The track is an audio track, as defined in IEC 908.
3-254	Reserved for future standardisation.
255	The volume associated with the Volume Space Table containing this Track Specification Record shall have unallocated sectors, and this Track Specification Record shall identify the unallocated sectors of the volume. The Session Number field, the Track Number field, Track Type field and Track Flags field shall be set to 0.

10.6.5 Track Flags (RBP 11)

This field shall specify certain attributes of the track as shown in table 3/18.

If the value of the Track Contents field is not 1, this field shall be set to 0.

Table 18 - Track Flags interpretation

Bit	Interpretation
0	If set to ZERO, the track shall not contain any End Transaction Descriptor. If set to ONE, the track may contain End Transaction Descriptors.
1	If set to ZERO, the track is not the assigned End Transaction Track. If set to ONE, the track is the assigned End Transaction Track.
2-7	Reserved for future standardisation and shall be set to ZERO.

If bit 1 is set to ONE, bit 0 shall be set to ONE.

10.6.6 Packet Size (RBP 12)

If bit 9 and bit 10 of the Track Type field are ONE, this field shall specify the length in data sectors of each packet of the track. Otherwise, this field shall be set to #00.

10.6.7 Start Location of Track (RBP 20)

If the value in the Track Contents field is not 255, this field shall specify the sector number of the first sector allocated to the track identified by the Track Specification Record.

If the value in the Track Contents field is 255, this field shall specify the lowest sector number of the unallocated sectors (see 3/9.1.5).

10.6.8 End Location of Track (RBP 28)

If the value in the Track Contents field is not 255, this field shall specify the sector number of the last sector of the track identified by the Track Specification Record.

If the value in the Track Contents field is 255, this field shall specify the greatest sector number of the volume allowed by the standard for recording.

10.6.9 Last Written Sector (RBP 36)

If the value in the Track Contents field is 1, this field shall specify the sector number of the greatest numbered sector of the track containing information recorded according to this Part.

If the value in the Track Contents field is not 1, this field shall be set to 0.

11 File set structure

11.1 File set

One or more file sets shall be recorded over a volume set. Each file set shall have an associated path table (see 3/13.3) which identifies every directory in the directory hierarchy (see 3/13.1) describing the set of files in the file set.

The prevailing File System Descriptor Set (see 3/11.2.2) of a volume set shall identify each file set in the volume set. A file set shall be identified by a File Set Descriptor (see 3/12.1) recorded in the prevailing File System Descriptor Set. The File Set Descriptor of a file set shall specify:

- the file set identification consisting of the File Set Character Set and the File Set Identifier fields.
- the set of characters allowed in certain fields of the descriptors associated with the file set.
- certain attributes of the file set.

11.2 File System Descriptor Set

A File System Descriptor Set shall be recorded according to the schema shown in figure 3/3.

```
[File System Descriptor Set]{
      <File Set Descriptor>1+
      <Implementation Use Descriptor>0+
      <Terminating Descriptor>1+
}
```

Figure3 - File System Descriptor Set schema

A File System Descriptor Set shall contain one or more File Set Descriptors (see 3/12.1). A File Set Descriptor shall specify a file set and certain attributes of the file set. Each File Set Descriptor of a File System Descriptor Set shall have an assigned File Set Descriptor Sequence Number. The set of all File Set Descriptor Sequence Numbers in a File System Descriptor Set shall form a continuous ascending sequence of integers starting from 1. All File Set Descriptors with identical File Set Descriptor Sequence Number shall have identical contents. No File Set Descriptor shall specify the same file set identification as any other File Set Descriptor in the prevailing File System Descriptor Set (see 3/11.2.2).

A File System Descriptor Set may contain zero or more Implementation Use Descriptors (see 3/12.2). An Implementation Use Descriptor shall identify an implementation and contain information for that implementation's use.

The sequence shall be terminated by a sequence of one or more identical Terminating Descriptors (see 3/10.5).

11.2.1 Recording of File System Descriptor Sets

The descriptors of a File System Descriptor Set shall be recorded in a sequence of sectors having consecutive ascending addresses (see 3/9.3.1). The location of a File System Descriptor Set shall be the location of the first descriptor in the File System Descriptor Set. A File System Descriptor Set shall end in the track in which it begins.

One or more File System Descriptor Sets shall be recorded on a volume set.

11.2.2 Prevailing File System Descriptor Set

The prevailing File System Descriptor Set of a volume set shall be the File System Descriptor Set identified by the prevailing End Transaction Descriptor of the volume set (see 3/9.6.3).

The information in the prevailing File System Descriptor Set shall override the information in all other File System Descriptor Sets of the volume set recorded on those volumes whose volume sequence numbers are less than, or equal to, the assigned volume set size of the volume on which the prevailing File System Descriptor Set is recorded.

The prevailing File System Descriptor Set of a volume set shall be recorded on the volume with the greatest volume sequence number of the volume set.

12 File set data structures

12.1 File Set Descriptor

The File Set Descriptor shall identify a set of files and directories and shall be recorded in the format shown in table 3/19.

Table 19 - File Set Descriptor format

BP	Length	Name	Contents
0	1	Structure Type	`Uint8` (1/7.1.1) = 3
1	5	Standard Identifier	bytes = “CDW02”
6	1	Structure Version	`Uint8` (1/7.1.1) = 2
7	1	File Structure Version	`Uint8` (1/7.1.1) = 2
8	64	Descriptor Character Set	`charspec` (1/7.2.1)
72	64	File Set Character Set	`charspec` (1/7.2.1)
136	32	File Set Identifier	`dstring` (1/7.2.12)
168	4	File Set Descriptor Sequence Number	`Uint16BOTH` (3/8.3)
172	4	Control Flags	`Uint32` (1/7.1.5)
176	4	Interchange Level	`Uint16BOTH` (3/8.3)
180	4	Maximum Interchange Level	`Uint16BOTH` (3/8.3)
184	4	Maximum Character Set List	`Uint32` (1/7.1.5)
188	32	Domain Identifier	`regid` (1/7.4)
220	12	File Set Creation Date and Time	`timestamp` (1/7.3.1)
232	12	File Set Expiration Date and Time	`timestamp` (1/7.3.1)
244	12	File Set Effective Date and Time	`timestamp` (1/7.3.1)
256	130	Application Identifier	`dstring` (1/7.2.12)
386	130	Publisher Identifier	`dstring` (1/7.2.12)
516	130	Data Preparer Identifier	`dstring` (1/7.2.12)
646	130	Copyright File Identifier	`dstring` (1/7.2.12)
776	130	Abstract File Identifier	`dstring` (1/7.2.12)
906	130	Bibliographic File Identifier	`dstring` (1/7.2.12)
1 036	512	Application Use	bytes
1 548	500	Reserved	#00 bytes

12.1.1 Structure Type (BP 0)

The number in this field shall specify 3.

12.1.2 Standard Identifier (BP 1)

This field shall specify “CDW02”.

12.1.3 Structure Version (BP 6)

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

12.1.4 File Structure Version (BP 7)

This field shall specify the version of the specification of the records of a directory and of a path table which describe the directory hierarchy (see 3/13.1) associated with the file set. The value 2 shall indicate the structure of this Part.

12.1.5 Descriptor Character Set (BP 8)

This field shall specify the d-characters (see 1/7.2) allowed in the Application Identifier, Publisher Identifier, and Data Preparer Identifier fields if the value of the first byte of these fields is #2B.

12.1.6 File Set Character Set (BP 72)

This field shall specify the d-characters (see 1/7.2) allowed in the following fields: File Set Identifier, Copyright File Identifier, Abstract File Identifier, and Bibliographic File Identifier. If the first byte of the following fields is #2D, it shall also specify the d-characters (see 1/7.2) allowed in these fields: Application Identifier, Publisher Identifier, and Data Preparer Identifier fields. In addition, it shall specify the d-characters (see 1/7.2) allowed in a pathname (see 3/13.4), and the File Identifier fields of all of the Path Table Records (see 3/15.2.9) and Directory Records (see 3/15.1.10) of the file set.

12.1.7 File Set Identifier (BP 136)

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

12.1.8 File Set Descriptor Sequence Number (BP 168)

This field shall specify the assigned File Set Descriptor Sequence Number of this descriptor.

12.1.9 Control Flags (BP 172)

This field shall specify the control attributes of the File Set Descriptor as shown in table 3/20.

Table 20 - Control Flags interpretation

Bit	Interpretation
0	If set to ONE, shall mean that each File System Descriptor Set recorded after the recording of this File Set Descriptor shall contain a File Set Descriptor with contents identical to this File Set Descriptor. If set to ZERO, shall mean that no such restriction is specified.
1-31	Reserved for future standardisation and shall be set to ZERO.

12.1.10 Interchange Level (BP 176)

This field shall specify the current level of medium interchange (see 3/16) of the file set described by this descriptor.

12.1.11 Maximum Interchange Level (BP 180)

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

12.1.12 Maximum Character Set List (BP 184)

The Descriptor Character Set and File Set Character Set fields in this descriptor shall not specify a character set not specified by the Maximum Character Set List field.

The maximum character set list shall be a proper subset of the maximum character set list (see 3/10.1.17) specified in the Primary Volume Descriptor of the prevailing Volume Descriptor Set in the volume having the greatest volume sequence number of the volume set.

12.1.13 Domain Identifier (BP 188)

This field shall specify an identification of a domain which shall specify rules on the use of, and restrictions on, certain fields in the descriptors associated with the file set 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` (see 1/7.4) shall include all information recorded in the file set described by this File Set Descriptor.

12.1.14 File Set Creation Date and Time (BP 220)

This field shall specify the date and time of the day of recording the End Transaction Descriptor that identifies the first transaction associated with the file set. No such date and time shall be specified prior to the recording of the first End Transaction Descriptor on the volume on which the File Set Descriptor is recorded.

12.1.15 File Set Expiration Date and Time (BP 232)

This field shall specify the date and time of the day at which the information in the file set may be regarded as obsolete. If the date and time are not specified, then the information shall not be regarded as obsolete.

12.1.16 File Set Effective Date and Time (BP 244)

This field shall specify the date and time of the day at which the information in the file set may be used. If the date and time are not specified, then the information may be used at once.

12.1.17 Application Identifier (BP 256)

If the first byte of this field is set to #2B, the remaining bytes of this field shall specify an identification of an application for the file set. The character set of this field shall be specified by the Descriptor Character Set field (see 3/12.1.5).

If the first byte is set to #2D, the remaining bytes of this field shall specify a File Identifier (see 3/13.7.1) for a file containing an identification of the application. The file shall be described in the root directory of the directory hierarchy associated with the file set. The file version number (see 3/13.7.2)of the file shall specify 1. The character set of this field shall be specified by the File Set Character Set field (see 3/12.1.6).

If this field contains all #00 bytes, then no such application is specified.

12.1.18 Publisher Identifier (BP 386)

If the first byte of this field is set to #2B, the remaining bytes of this field shall specify an identification of the user who specified what shall be recorded in the file set. This user is referred to as the publisher. The character set of this field shall be specified by the Descriptor Character Set field (see 3/12.1.5).

If the first byte is set to #2D, the remaining bytes of this field shall specify a File Identifier (see 3/13.7.1) for a file containing an identification of the publisher. The file shall be described in the root directory of the directory hierarchy associated with the file set. The file version number (see 3/113.7.2) of the file shall be 1. The character set of this field shall be specified by the File Set Character Set field (see 3/12.1.6).

If this field contains all #00 bytes, then no such user is specified.

12.1.19 Data Preparer Identifier (BP 516)

If the first byte of this field is set to #2B, the remaining bytes of this field shall specify an identification of the person or other entity which controls the preparation of the data to be recorded in the file set. This entity is referred to as the data preparer. The character set of this field shall be specified by the Descriptor Character Set field (see 3/12.1.5).

If the first byte is set to #2D, the remaining bytes of this field shall specify a File Identifier (see 3/13.7.1) for a file containing an identification of the data preparer. The file shall be described in the root directory of the directory hierarchy associated with the file set. The file version number (see 3/13.7.2) of the file shall be 1. The character set of this field shall be specified by the File Set Character Set field (see 3/12.1.6).

If this field contains all #00 bytes, then no such data preparer is specified.

12.1.20 Copyright File Identifier (BP 646)

This field shall specify the File Identifier (see 3/13.7.1) of a file in the root directory of the directory hierarchy associated with the file set. The file shall be interpreted as containing a copyright statement that shall be applied to the files of the file set. The file version number (see 3/13.7.2) of the file shall be 1. The character set of this field shall be specified by the File Set Character Set field (see 3/12.1.6).

If this field contains all #00 bytes, then no such file is specified.

12.1.21 Abstract File Identifier (BP 776)

This field shall specify the File Identifier (see 3/13.7.1) of a file in the root directory of the directory hierarchy associated with the file set. The file shall be interpreted as containing an abstract statement that applies to the files of the file set. The file version number (see 3/13.7.2) of the file shall be 1. The character set of this field shall be specified by the File Set Character Set field (see 3/12.1.6).

If this field contains all #00 bytes, then no such file is specified.

12.1.22 Bibliographic File Identifier (BP 906)

This field shall specify the File Identifier (see 3/13.7.1) of a file in the root directory of the directory hierarchy associated with the file set. The file shall be interpreted as containing bibliographic records and shall be interpreted according to standards that are the subject of an agreement between the originator and the recipient of the file set. The file version number (see 3/13.7.2) of the file shall be 1. The character set of this field shall be specified by the File Set Character Set field (see 3/12.1.6).

If this field contains all #00 bytes, then no such file is specified.

12.1.23 Application Use (BP 1 036)

This field shall be used to store application use extended attributes (see 3/14). Application use extended attributes shall be recorded contiguously according to the extended attribute format (see 3/14.2). Any remaining bytes in this field shall be set to #00.

If this field contains all #00 bytes, then no such extended attributes are specified.

NOTE

The "CE" extended attribute may be recorded in the Application Use field to provide an additional extended attribute area to store more application use extended attributes.

12.1.24 Reserved (BP 1 548)

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

12.2 Implementation Use Descriptor

The Implementation Use Descriptor shall identify an implementation which can recognise and act upon the contents of the Implementation Use field in this descriptor. It shall be recorded in the format shown in table 3/21.

Table 21 - Implementation Use Descriptor format

BP	Length	Name	Contents
0	1	Structure Type	`Uint8` (1/7.1.1) = 4
1	5	Standard Identifier	bytes = "CDW02"
6	1	Structure Version	`Uint8` (1/7.1.1) = 2
7	65	Reserved	#00 byte
72	32	Implementation Identifier	`regid` (1/7.4)
104	1 944	Implementation Use	bytes

12.2.1 Structure Type (BP 0)

The number in this field shall specify 4.

12.2.2 Standard Identifier (BP 1)

This field shall specify "CDW02".

12.2.3 Structure Version (BP 6)

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

12.2.4 Reserved (BP 7)

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

12.2.5 Implementation Identifier (BP 72)

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.

12.2.6 Implementation Use (BP 104)

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

13 Directory and file structures

13.1 Directory hierarchy

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 identifying another directory by other than an alias file (see 3/13.9) shall be called the 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 hierarchy shall be described by a path table (see 3/13.3). Each directory shall be identified as being either an ECMA-119 directory (see 3/13.1.1) or an ECMA-168 directory (see 3/13.1.2) by the value of the Version bit of the Files Flag field (see 3/15.2.5) of the Path Table Record (see 3/15.2) describing the directory.

13.1.1 ECMA-119 directories

A directory identified as an ECMA-119 directory shall contain only Directory Records according to 9.1 of ECMA-119 and the Multi-Extent bit of the File Flags field (see 9.1.6 of ECMA-119) shall be set to ZERO in all Directory Records of the directory.

The Directory Records of each ECMA-119 directory shall be ordered according to 3/13.1.1.2 and recorded according to 3/13.2.

NOTE

When the Version bit of the Files Flag field (see 3/15.2.5) of the Path Table Record (see 3/15.2) describing a directory is set to ZERO, this Part requires that all Directory Records of the directory be interpreted according to ECMA-119, instead of this Part. However, this Part requires that any file with more than one file section be identified with a Part 3 Directory Record with both the Version bit and the Multi-Extent bit of the File Flags field (see 3/15.1.5) set to ONE and the “MF” extended attributes (see 3/15.3.10) used to describe the file sections of the file. This means that if the same files are to be read by both an ECMA-119 conforming receiving system and a Part 3 receiving system , there would need to be separate directories recorded to describe the same files. One directory, identified by a Part 3 path table, would use the “MF” extended attribute to describe files with more than one file section. The other directory, identified by an ECMA-119 path table (not used by this Part, would conform to ECMA-119 and use multiple Directory Records to describe the same file.

NOTE

In an ECMA-119 directory, neither a file nor a subdirectory may be identified by an alias file.

13.1.1.1 Directory descriptors in an ECMA-119 directory

A directory identified as an ECMA-119 directory shall contain a set of directory descriptors, each of which identifies a component of the directory. A component shall be a file or a subdirectory or the parent directory of the directory or the directory itself.

Each directory descriptor shall specify:

– the name of a component of the directory. The length, in bytes, of the name of each component shall be greater than 0.

- whether the identified component is a directory.
- the location and length, in bytes, of the identified component.

NOTE

This Part does not permit the starting logical block number of a file section to be 0 (see 3/13.5.1.2).ECMA-119 does not have this restriction.

NOTE 1

This Part requires that the length of all but the last file section of a file be an integer multiple of the logical block size (see3/13.5.1). ECMA-119 does not have this restriction.

- the attributes of the identified component. If the component is identified as a directory, the attributes specified by the Directory Record shall not conflict with the attributes of the directory specified by the Path Table Record identifying the directory.

For the descriptors in an ECMA-119 directory

- with the exception of Directory Records identifying associated files (see 3/13.8), there shall not be more than one File Identifier (see 7.5 of ECMA-119) with the same File Name (see 7.5.1 of ECMA-119) and File Version Number (see 7.5.1 of ECMA-119).
- there shall be one Directory Record identifying each subdirectory of the directory.
- the path table describing the directory hierarchy of which the directory is member shall not identify any subdirectory of the directory with a File Identifier that is the same as the File Name of any Directory Record of the directory, unless the Directory Record is identifying the same subdirectory.
- there shall be exactly one Directory Record identifying the parent directory.
- there shall be exactly one Directory Record identifying the directory itself.

NOTE

The character set specifying the characters (see 7.4 of ECMA-119) used in the directory descriptors is specified in the File Set Descriptor for the directory hierarchy of which the directory is a member.

13.1.1.2 Order of Directory Records in an ECMA-119 directory

A directory identified as containing an ECMA-119 directory shall be ordered as follows.

The first Directory Record of each ECMA-119 directory shall describe the directory in which it is recorded and shall have a File Identifier consisting of a single #00 byte (see 6.8.2.2 of ECMA-119).

The second Directory Record of each ECMA-119 directory shall describe the parent directory for the directory in which it is recorded and shall have a File Identifier consisting of a single #01 byte (see 6.8.2.2 of ECMA-119).

If there are more than two Directory Records in an ECMA-119 directory, the remaining records shall be ordered according to 9.3 of ECMA-119.

NOTE

This makes it possible to import directories described by ECMA-119 into the directory hierarchy of a file set without format conversion, if certain restrictions as specified by this Part are met and the medium is recorded according to ECMA-119 and this Part. Each ECMA-119 directory will have a Directory Record identifying the parent of the directory and a Directory record identifying the directory itself. Part 3 directories will not contain either of these directory descriptors.

13.1.1.3 Interpretation of ECMA-119 Extended Attribute Records

If a Directory Record of a directory identified as an ECMA-119 directory specifies that an Extended Attribute Record (see 9.5 of ECMA-119) is recorded for the file, the fields of the Extended Attribute Record shall be interpreted as specified in 9.5 of ECMA-119 except as follows:

- The Owner Identification field shall be interpreted according to 3/15.3.4.4.
- The Group Identification field shall be interpreted according to 3/15.3.4.5.

- The Permissions field shall be interpreted according to 3/15.3.4.6.

NOTE

This assures a consistent interpretation of these three fields for both types of directories that may be included in a Part 3 directory hierarchy.

13.1.2 Part 3 directories

A directory identified as a Part 3 directory shall contain only Directory Records according to 3/15.1.

The Directory Records of each Part 3 directory shall be ordered according to 3/13.1.2.2 and shall be recorded according to 3/13.2.

13.1.2.1 Directory descriptors in an ISO/IEC 13490-2 directory

A directory identified as an ISO/IEC 13490-2 directory shall contain a set of directory descriptors, each of which identifies a component file or a component subdirectory. A directory descriptor that identifies a component file or subdirectory by identifying the location of each file section (see 3/13.5.1) for that component shall be recorded as a Directory Record (see 3/15.1) with the Alias bit of the File Flags field (see 3/15.1.5) set to ZERO. A directory descriptor that identifies a component file or subdirectory of the directory by specifying the pathname (see 3/13.4) of the actual file or directory shall be referred to as an alias and shall be recorded as a Directory Record (see 3/15.1) with the Alias bit of the File Flags field (see 3/15.1.5) set to ONE.

Each directory descriptor shall specify:

- the name of a component of the directory. The length, in bytes, of the name of each component shall be greater than 0.
- whether the identified component is a directory. When the descriptor identifies an alias, this indication is contained in the directory descriptor for the file identified by the pathname specified by the alias.
- the location and length, in bytes, of the identified component.

NOTE

This Part does not permit the starting logical block number of a file section to be 0 (see 3/13.5.1.2). ECMA-119 does not have this restriction.

NOTE

This Part requires that the length of all but the last file section of a file be an integer multiple of the logical block size (see 3/13.5.1). ECMA-119 does not have this restriction.

- the attributes of the identified component. If the component is identified as a directory, the attributes specified by the Directory Record shall not conflict with the attributes of the directory specified by the Path Table Record identifying the directory.

For the descriptors in a Part 3 directory

- with the exception of Directory Records (see 3/15.1) identifying associated files (see 3/13.8), there shall not be more than one descriptor with the same File Identifier (see 3/15.1.10) and File Version Number (see 3/15.1.13).
- there need not be any Directory Record identifying any subdirectory of the directory.
- the path table describing the directory hierarchy of which the directory is member shall not identify any subdirectory of the directory with a File Identifier that is the same as the File Identifier of any Directory Record of the directory, unless the Directory Record is identifying the same subdirectory.
- a descriptor identifying a directory shall have a File Version Number of 1.

A Directory Record specifying a file in which a directory is recorded shall not specify a “CS” Extended Attribute.

NOTE

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

13.1.2.2 Order of Directory Records in a Part 3 directory

A directory identified as containing a Part 3 directory shall be ordered according to the following criteria in descending order of significance:

1) In ascending order according to the relative value of the 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 the File Version Number field (see 3/15.1.13).

3) In descending order according to the value of the Associated File bit of the File Flags field (see 3/15.1.5).

NOTE

The order of the Directory Records in a Part 3 directory is independent of the charspec (see 1/7.2.1) applying to the directory because the File Identifiers are sorted as if they were binary values.

13.1.3 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}.

13.1.4 Depth of a directory hierarchy

A directory hierarchy shall contain one or more levels. The root directory shall be the only directory at level 1 of the directory hierarchy. If a directory is at level *m* of the directory hierarchy, its subdirectories shall be at level *m+1*.

The depth of a directory hierarchy shall be the number of levels in the directory hierarchy.

13.2 Recording of directory files

A directory shall be recorded as the contents of a directory file according to the schema shown in figure 3/4.

```
[directory file] {
      [data sector] {
            <Directory Record>0+
            <#00 byte>0+
      }0+
}
```

Figure 4 - Directory file schema

The first or only Directory Record recorded in a data sector (see 3/9.1.1.3) shall begin at the first byte of that sector. Each subsequent Directory Record recorded in that sector shall begin at the byte immediately following the last byte of the preceding Directory Record in that sector. Each Directory Record shall end in the sector in which it begins. Unused byte positions after the last Directory Record in a sector shall be set to #00.

A directory file shall be recorded in a single extent. The first logical block of the extent shall have a logical block number which is the lowest logical block number in the sector that contains the logical block.

A directory file shall not be recorded in interleaved mode (see 3/13.5.1.1.1).

13.2.1 Directory file length

The length, in bytes, of a directory file shall be the sum of

– the lengths, in bytes, of all Directory Records in the directory;

– the number of unused byte positions after the last Directory Record in all sectors in which the directory file is recorded.

13.3 Path Table

A path table shall contain a set of Path Table Records (see 3/15.2) describing the directory hierarchy of a file set.

For each directory in a directory hierarchy, the associated path table shall contain one Path Table Record which identifies the directory, its parent directory, its location and its size. The records in a path table shall be numbered starting from 1. This number shall be referred to as the directory number. The first record in the path table shall identify the root directory and its location.

The directory number of a directory shall be the ordinal number of the Path Table Record that identifies the directory.

Path tables shall be recorded on the volume of a volume set having the greatest assigned volume sequence number and shall override the path tables recorded on volumes of the volume set having a lower assigned volume sequence number.

A Path Table Record shall not specify any “FC”, “CS”, “RF”, or “MF” extended attributes (see 3/14.3).

The File Identifier field of a Path Table Record identifying the root directory of a directory hierarchy shall consist of a single #00 byte.

13.3.1 Order of Path Table Records

The records in a path table shall be ordered by the following criteria in descending order of significance:

1) in ascending order according to the level of the identified directory in the directory hierarchy;

2) in ascending order according to the directory number of the parent directory of the identified directory;

3) in ascending order according to the relative value of the File Identifier of the identified directory, where File Identifiers shall be valued as follows:

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

   – If 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.

NOTE

The order is independent of the charspec (see 1/7.2.1) applying to the path table because the File Identifiers are sorted as if they were binary values.

13.3.2 Recording of Path Tables

A path table shall be recorded as the contents of a path table file according to the schema shown in figure 3/5.

```
[path table file] {
      <path table record>1+
}
```

Figure 5 - Path Table schema

The first Path Table Record shall be recorded starting at the first byte of the path table file. Each subsequent Path Table Record shall begin at the byte immediately following the last byte of the preceding Path Table Record.

The length of a path table file shall be the sum of the lengths of all Path Table Records recorded in the file.

The path table file shall be recorded in a single extent, and shall not be recorded in the interleaved mode (see 3/13.5.1.1.1).

13.3.3 Identification of Path Tables

Path tables shall be identified by the Path Tables Information field (see 3/10.4.17) of the prevailing End Transaction Descriptor of the volume set (see 3/9.6.3). There shall be one path table for each file set (see 3/11.1) identified by the prevailing File System Descriptor Set (see 3/11.2) of the volume set. Each path table shall be recorded as the contents of a separate file as specified in 3/8.3.

A Directory Record (see 3/15.1) shall identify each such file as follows:

– the File Identifier field of the Directory Record shall contain the sequence number of the File Set Descriptor (see 3/12.1) identifying the file set. This field shall be a 4-byte field recorded as a `Uint16BOTH` (see 3/8.3).

– the Length of File Identifier field of the Directory Record shall be set to 4.

– the File Version Number field of the Directory Record shall be set to 1.

These files shall not be identified by any directory (see 3/13) of any file set (see 3/11.1) recorded in the volume set.

13.3.4 Path Tables Location Directory

The Path Tables Location Directory shall be recorded as a directory (see 3/13.1). The directory shall identify, as specified in 3/13.3.3, a file for each path table.

A Path Table Record (see 3/15.2) shall identify the directory as follows:

– the File Identifier field of the Path Table Record shall contain 0. This field shall be a 4-byte field recorded as a `Uint16BOTH` (see 3/8.3).

– the Length of File Identifier field of the Path Table Record shall be set to 4.

– the Parent Directory Number field of the Path Table Record shall be set to 0.

The Path Tables Location Directory need not be recorded if there is only one file set identified by the prevailing File System Descriptor Set of the volume set.

The Path Tables Location Directory shall not be identified by any path table (see 3/13.3) or by any directory (see 3/13) of any file set (see 3/11.1) recorded in the volume set.

13.4 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 3/15.4) 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.

13.4.1 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 Component with a Component Type of 2 (see 3/15.4.1).

– each other component is a Path Component with a Component Type of 5 and is not an alias.

The length of a resolved pathname shall be the sum of the following:

– the value of the Length of Component Identifier field (see 3/15.4.2) for each component.

– the number of components.

NOTE

The resolved pathname is mainly used in 3/16. 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 sets.

Note that the length of the resolved pathname does not provide for the length of the file version number (see 3/13,7,2) associated with the final component of the pathname.

13.5 Files

A file shall be described by a Directory Record (see 3/15.1). The Directory Record shall specify the attributes of the file and shall identify all the file sections of the file. Some of the attributes shall be recorded in the Directory Record itself; the remainder shall be recorded as extended attributes (see 3/14).

The data of a file shall be recorded in one or more file sections (see 3/15.5.1). The file sections of a file may be located on different volumes.

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

13.5.1 File sections

That part of a file recorded in an extent or in the File Contents field of an “FC” extended attribute (see 3/15.3.8) is referred to as a file section.

If a file contains only one file section, the data of the file may be recorded using the “FC” extended attribute. If a file contains more than one file section, each file section of the file shall be recorded in an extent.

NOTE

The “FC” extended attribute may be used to record the contents of a small file in the Directory Record identifying the file. This results in better performance and better utilisation of disk space.

The number of bytes in a file section shall be referred to as the length of the file section. The length of each, except the last, file section of a file shall be an integer multiple of the logical block size. The length of the last file section of a file need not be an integer multiple of the logical block size. If there are no bytes recorded in the file section, the length shall be 0.

NOTE

ECMA-119 does not require that the length of any file section be an integer multiple of the logical block size,

13.5.1.1 Mode of recording a file section

A file section may be recorded in an extent, or in an “FC” extended attribute (see 3/15.3.8).

If a file section is recorded in an extent, it shall be recorded either in interleaved mode (see 3/13.5.1.1.1) or in non-interleaved mode (see 3/13.5.1.1.2). All file sections of a file shall be recorded in the same mode.

13.5.1.1.1 Interleaved mode

A file section recorded in interleaved mode shall be recorded over a sequence of file units in an extent according to the schema shown in figure 3/6 starting with the first byte of the first logical block of the extent.

```
[Interleaved Mode Extent] {
      <File Unit>
      {
            <Interleave Gap>
            <File Unit>
      } 0+
      <#00 byte> 0+
} 0+1
```

Figure 6 - Interleaved Mode Extent schema

Each file unit of the sequence shall consist of the same number of logical blocks, the logical block numbers of which shall form a continuous ascending sequence. The number of logical blocks in the file unit shall be the assigned file unit size for the extent. The first logical block of each file unit of the extent shall have a logical block number which is the lowest logical block number in the sector that contains the logical block.

An interleave gap shall start with the first logical block after the last logical block of a file unit of the extent and end with the greatest numbered logical block prior to the first logical block of the next, if any, file unit of the extent. The number of logical blocks in the interleave gap shall be the assigned interleave gap size. This number shall be 0 if there are no logical blocks in the interleave gap. The interleave gap size shall be the same for all interleave gaps of an extent.

NOTE

Interleaved mode allows the non-contiguous recording of a file section. This may permit faster access to the data.

13.5.1.1.2 Non-interleaved mode

A file section recorded in an extent in non-interleaved mode shall be recorded over a sequence of logical blocks in the extent.

13.5.1.2 Identification of file sections

A file section recorded in an extent shall be identified by the Directory Record describing the file section by specifying the sequence number of the volume of the volume set, the logical block number on that volume at which the first byte of the file section is recorded and the length of the file section..

Logical block number 0 shall not be the starting logical block number of a file section.

NOTE

ECMA-119 permits the starting logical block number of a file section to be 0.

Any file section that is identified as beginning with a logical block number of #FFFFFFFF shall not have any recorded logical blocks and shall be treated as if the file section consisted of the number of #00 bytes specified by the Directory Record describing the file section.

A file section shall be identified as being recorded in the File Contents field of an “FC” extended attribute (see 3/15.3.8) by recording the extended attribute in the standard extended attribute area (see 3/14.1) of the file.

13.5.1.3 Ordering of file sections

If a file consists of more than one file section, each of the file sections shall be identified by a File Section Record (see 3/15.5) in the File Sections field of the “MF” extended attribute (see 3/15.3.10) identified by the Directory Record that describes the file.

The File Section Record starting at the first byte of the File Sections field of the “MF” extended attribute shall describe the first file section of the file. Each, if any, successive File Section Record of the File Sections field in the “MF” extended attribute shall specify the next file section of the file.

13.5.1.4 Recording of a file section

All file sections of a file shall be recorded in tracks that are assigned with the same track type (see 3/10.6.3). A file section shall end in the track in which it begins.

If a file section is recorded in a track with fixed-length packets, the file section need not end in the packet in which it begins.

If a file section is recorded in a track with variable-length packets, the file section shall end in the packet in which it begins.

If a file section is recorded in interleaved mode with either fixed-length or variable-length packets, each file unit of the file section shall end in the packet in which it begins.

13.5.1.5 Data space of a file section

13.5.1.5.1 Interleaved mode

If a file section is recorded in an extent in interleaved mode, the set of the file units in which the successive parts of the file section are recorded shall be the data space of the file section.

The bytes in the data space of a file section shall be numbered consecutively. The numbering shall start from 0 which shall be assigned to the first byte of the first logical block of the first file unit, if any, of the data space of the file section. The numbering shall continue through successive bytes of that logical block, and then through successive bytes of each successive logical block, if any, of the first file unit, and then through successive bytes of the logical block(s) of each successive file unit, if any, of data space of the file section.

The numbering shall end with a number equal to 1 less than the number of bytes of the file recorded in the data space of the file section; or this number shall equal 0 if there are no bytes of the file recorded in the data space of the file section.

13.5.1.5.2 Non-interleaved mode

If a file section is recorded in an extent in non-interleaved mode, the set of logical blocks over which the file section is recorded shall be the data space of the file section.

The bytes in the data space of the file section shall be numbered consecutively. The numbering shall start from 0 which shall be assigned to the first byte of the first logical block, if any, of the data space of the file section. The numbering shall continue through successive bytes of that logical block, and then through successive bytes of each successive logical block, if any, of the data space of the file section.

The numbering shall end with a number equal to 1 less than the number of bytes of the file recorded in the data space of the file section; or this number shall equal to 0 if there are no bytes of the file recorded in the data space of the file section.

13.5.1.5.3 “FC” Extended Attribute

If a file is recorded in an “FC” extended attribute, the file section shall be recorded in the File Contents field (see 3/15.3.8.1.4 and 3/15.3.8.2.5) of the “FC” extended attribute which shall be the data space of the file section.

The bytes in the File Contents field shall be numbered consecutively. The numbering shall start from 0 which shall be assigned to the first byte of the File Contents field and the numbering shall continue through successive bytes of the File Contents field. The numbering shall end with a number equal to 1 less than the size of the file; or this number shall equal 0 if there are no bytes of the file recorded in the data space of the file section.

13.5.2 Data space of a file

The data space of a file shall be the ordered set of file sections on which the data of the file is recorded.

The bytes in the data space of a file shall be numbered with consecutive integers in an ascending sequence. The numbering shall start from 0 which shall be assigned to the first byte of the first, if any, file section of the file. The numbering shall continue through successive bytes of that file section, and then through successive bytes of each successive file section, if any, of the file.

The numbering shall end with a number equal to 1 less than the sum of the number of bytes in all file sections of the file; or this number shall equal 0 if there are no bytes of the file recorded in the data space of the file.

13.6 Record structure

The information in a file may be organised as a set of records according to 3/15.3.5. The structure and attributes of these records are specified in ECMA-167, Part 5. For the purposes of ECMA-167, Part 5, the data space of a file is specified by 3/13.5.2 and

– if the "CS" extended attribute (see 3/15.3.2) is specified for the file, then that extended attribute specifies how the bytes of the file shall be interpreted as characters,

– if the "CS" 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 apply only for the purposes of partitioning the file's contents into records, and need not apply to the contents of the records.

NOTE

Some record formats (see ECMA-167, Part 5) specify records delimited by a specific character sequence.

13.7 File identification

A file shall be identified by a file identifier (see 3/13.7.1) and an associated file version number (see 3/13.7.2).

13.7.1 File identifier

Except as specified in 3/9.5.3, 3/9.5.4, 3/13.1.1.2, 3/13.3.3 and 3/133.4, a file identifier shall consist of a sequence of one or more d-characters (see 1/7.2). The length of a file identifier shall be the number of bytes comprising the file identifier. The length of a file identifier shall not exceed 255 bytes.

13.7.2 File version number

A file version number shall be a number in the range 1 to 32 767 inclusive. The numbers 32 768 to 65 535 inclusive are reserved for future standardisation.

13.8 Associated file

An associated file has a relationship not specified by this Part to another file that has been assigned the same file identifier and file version number (see 3/13.7.2) as that of the associated file in the same directory.

13.9 Alias File

An alias file is a file that contains the pathname (see 3/13.4) of a directory or another file.

14 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, a directory, a Primary Volume Descriptor, a Supplementary Volume Descriptor or a File Set Descriptor. The term "instances of an extended attribute" shall refer to all extended attributes recorded for the file (see 3/14.1), directory (see 3/14.1), Primary Volume Descriptor (see3/14.2), Supplementary Volume Descriptor (see 3/14.2) or File Set Descriptor (see 3/14.2) with identical contents of their Attribute Type and Attribute Subtype fields (see 3/15.3.1).

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:

– Standard Defined Extended Attributes are attributes with type 0 to 2 048 inclusive and 65 536. Attribute types 1 to 11 inclusive, 2 048 and 65 536 are registered according to ISO/IEC 13800 and are recorded according to 3/15.3. Attribute type 12 is registered according to ISO/IEC 13800 and shall not be recorded according to 3/14.1 or 3/14.2. Attribute types 13 to 2 047 are reserved for reserved for registration according to ISO/IEC 13800. Attribute type 0 is reserved for future standardisation by ISO/IEC 13800.

- Implementation Use Extended Attributes are attributes with type 2 049 to 65 535 inclusive which shall be registered according to ISO/IEC 13800, are recorded according to 3/15.3.1 and are reserved for implementation use according to ISO/IEC 13800.
- Application Use Extended Attributes are attributes with type 65 537 and above which shall be registered according to ISO/IEC 13800, are recorded according to 3/15.3.1 and are reserved for application use according to ISO/IEC 13800.

There shall be:

- zero or one instance of each attribute with type 1 through 11 inclusive.
- zero instances of each attribute with type 0 and 12.
- zero or more instances of each attribute with type 2 048 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` (see 1/7.4) recorded in the Implementation Identifier field of the attribute.
- 65 536: is specified by the `regid` (see 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.

The set of extended attributes associated with a file shall be recorded according to 3/14.1 in the Extended Attributes Area field of the Directory Record that identifies the file and, if any, Additional Extended Attributes Areas recorded according to 3/14.1.1 and identified by “CE” extended attributes (see 3/15.3.9).

The set of extended attributes associated with a directory shall be recorded according to 3/14.1 in the Extended Attributes Area field of the Path Table Record that identifies the directory and, if any, Additional Extended Attributes Areas recorded according to 3/14.1.1 and identified by “CE” extended attributes (see 3/15.3.9).

The set of extended attributes associated with a Primary Volume Descriptor or a Supplementary Volume Descriptor or a File Set Descriptor shall be recorded according to 3/14.2 in the Application Use field of that descriptor and, if any, Additional Extended Attributes Areas recorded according to 3/14.2.1 and identified by “CE” extended attributes (see 3/15.3.9).

14.1 Recording of extended attributes for files and directories

Extended attributes in the Extended Attribute Area field of a Directory Record and of a Path Table Record shall be recorded contiguously according to the schema shown in figure 3/7.

```
[Extended Attribute Area] {
      <Standard Defined Extended Attribute Existence>
      [extended attribute] {
            <Standard Defined Extended Attribute> |
            <Implementation Use Extended Attribute> |
            <Application Use Extended Attribute>
      } 0+
      <#00 byte> 0+
}
```

Figure 7 - Extended Attribute Area schema

The Standard Defined Extended Attribute Existence is specified in 3/15.1.11.

14.1.1 Recording of additional extended attributes for files and directories

Extended attributes in an additional extended attribute area (see 3/14 and 3/15.3.9) for files and directories shall be recorded contiguously in three non-overlapping areas within the additional extended attribute area according to the schema shown in figure 3/8. These areas shall be referred to as the Additional Standard Attribute Area, the Implementation Use Area and the Application Use Area.

```
[Additional Extended Attribute Area] {
      [Additional Standard Attribute Area] {
            <Standard Defined Extended Attribute>
            (other than a “CE” extended attribute) 0+
      }
      [Implementation Use Area] {
            <Implementation Use Extended Attribute> |
            <“PD” Extended Attribute>
      }  0+
      [Application Use Area] {
            <Application Use Extended Attribute> |
            <“PD” Extended Attribute>
      }  0+
      }
      <“CE” extended attribute>0+1
      <#00 byte> 0+
}
```

Figure 8 - Additional Extended Attribute Area schema for files and directories

14.2 Recording of extended attributes for a Primary Volume Descriptor or a Supplementary Volume Descriptor or a File Set Descriptor

Extended attributes in the Application Use field of a Primary Volume Descriptor and a Supplementary Volume Descriptor and a File Set Descriptor shall be recorded contiguously according to the schema shown in figure 3/9.

```
[Volume Structure Descriptor Application Use field] {
      {
            <Application Use Extended Attribute> |
            <“PD” Extended Attribute>
      }  0+
      <“CE” extended attribute>0+1
      <#00 byte> 0+
}
```

Figure 9 - Extended Attribute schema for a Primary Volume Descriptor or Supplementary Volume Descriptor or File Set Descriptor

14.2.1 Recording of additional extended attributes for a Primary Volume Descriptor or a Supplementary Volume Descriptor or a File Set Descriptor

Extended attributes in an additional extended attribute area (see 3/14 and 3/15.3.9) for a Primary Volume Descriptor and a Supplementary Volume Descriptor and a File Set Descriptor shall be recorded contiguously within the additional extended attribute area according to the schema shown in figure 3/10.

```
[Additional Extended Attribute Area] {
      {
            <Application Use Extended Attribute> |
            <“PD” Extended Attribute>
      }  0+
      <“CE” extended attribute>0+1
      <#00 byte> 0+
}
```

Figure 10 - Additional Extended Attribute Area schema for a Primary Volume Descriptor or a Supplementary Volume Descriptor or a File Set Descriptor

14.3 Extended attributes specified by this Part

The extended attributes specified in this Part are listed in table 3/22.

Table 22 - Extended attributes defined in this Part

Type	Name	Description
1	“CS”	Character Set Information (3/15.3.2)
2	“PD”	Padding (3/15.3.3)
3	“PM”	Alternate Permissions (3/15.3.4)
4	“RF”	Record Structure (3/15.3.5)
5	“TF”	File Times (3/15.3.6)
6	“TI”	Information Times (3/15.3.7)
7	“FC”	File Contents (3/15.3.8)
8	“CE”	Extended Attribute Area Continuation (3/15.3.9)
9	“MF”	File Section Description Area (3/15.3.10)
10	“UX”	ISO 9945-1 File Attributes (3/15.3.11)
11	“XR”	Previously recorded Directory Record (3/15.3.12)
2 048	“SU”	Implementation Use (3/15.3.13)
65 536	“AU”	Application Use (3/15.3.14)

NOTE

There need not be any extended attributes associated with a file, directory, Primary Volume Descriptor, Supplementary Volume Descriptor or File Set Descriptor.

NOTE

For performance reasons, often used extended attributes should be recorded in the Extended Attribute Area (see 3/14.1) of Directory Records and Path Table Records, rather than in the Additional Standard Attribute Area (see 3/14.1.1).

NOTE

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.

15 Directory and file data structures

15.1 Directory Record

A Directory Record shall be recorded in the format shown in table 3/23.

NOTE

The major differences between the Directory Record of this Part and the Directory Record specified in ECMA-119 are:

- *More attributes can be specified for a file.*
- *The file version number is a separate field, permitting easier implementation of file version numbers.*
- *Symbolic links are allowed.*

Table 23 - Directory Record

RBP	Length	Name	Contents
0	2	Length of Directory Record (=L_DR)	`Uint16` (1/7.1.3)
2	8	Location of File Section	`Uint32BOTH` (3/8.6)
10	8	Data Length	`Uint32BOTH` (3/8.6)
18	7	Recording Date and Time	`shorttimestamp` (1/7.3.2)
25	1	File Flags	`Uint8` (1/7.1.1)
26	1	File Unit Size	`Uint8` (1/7.1.1)
27	1	Interleave Gap Size	`Uint8` (1/7.1.1)
28	4	Volume Sequence Number	`Uint16BOTH` (3/8.3)
32	1	Length of File Identifier (=L_FI)	`Uint8` (1/7.1.1)
33	L_FI	File Identifier	bytes
[L_FI+33]	L_XAA	Extended Attribute Area	bytes
[L_XAA+L_FI+33]	*	Padding	#00 bytes
[*+L_XAA+L_FI+33]	4	File Version Number	`Uint16BOTH` (3/8.3)

15.1.1 Length of Directory Record (=L_DR) (RBP 0)

This field shall specify the length, in bytes, of the Directory Record.

15.1.2 Location of File Section (RBP 2)

If either the “MF” or “FC” extended attribute exists (see 3/15.1.11), this field shall be set to 0.

If neither the “MF” nor “FC” extended attribute exists, the file shall be recorded in one file section, and this field shall specify the logical block number of the logical block at which the first byte of the file section is recorded. The file section of the file shall be recorded in the volume with the volume sequence number specified in the Volume Sequence Number field.

If the value of this field is #FFFFFFFF, the value of the File Unit Size and Interleave Gap Size fields shall be 0, the Data Length field shall contain the size of the file, and the file shall be treated as if it consisted of all #00 bytes, and shall not be recorded.

15.1.3 Data Length (RBP 10)

If the “MF” extended attribute does not exist, the file shall contain only one file section and this field shall specify the length, in bytes, of the file section.

If the “MF” extended attribute exists, this field shall specify the sum of the lengths, in bytes, of all of the file sections described by the “MF” extended attribute.

15.1.4 Recording Date and Time (RBP 18)

This field shall specify the date and time of the day when this Directory Record is recorded.

15.1.5 File Flags (RBP 25)

This field shall specify certain characteristics of the file as shown in table 3/24.

Table 24 - File Flags interpretation

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 Directory Record does not identify a directory (see 3/13). If set to ONE, shall mean that: – the Directory Record identifies a directory. – the interpretation of any field in this Directory Record, other than the Length of Directory Record field and the File Flags field, is not specified by this Part (see 3/13.2.1). – the Associated File, Record, Alias and Multi-Extent bits shall be set to ZERO.
2	Associated File: If set to ZERO, shall mean that the file is not an associated file (see 3/13.8). If set to ONE, shall mean that the file is an associated file.
3	Record: If set to ZERO, shall mean there is no “RF” extended attribute (see3/15.3.5) for the file. If set to ONE, shall mean that a “RF” extended attribute is recorded for the file.
4	Protection: If set to ZERO, there is no “PM” extended attribute (see 3/15.3.4) for the file. If set to ONE, shall mean that a “PM” extended attribute is recorded for the file.
5	Version: This bit shall be set to the same value as the Version bit of the File Flags field (3/15.2.5) of the Path Table Record identifying the directory in which this Directory Record is recorded. If set to ZERO, shall mean that this Directory Record shall be interpreted according to3/13.1.1 and the Multi-Extent and Alias bits shall be set to 0. If set to ONE, shall mean that this Directory Record shall be interpreted according to3/13.1.2.
6	Alias: If set to ZERO, shall mean that the file is not an alias file (3/13.9). If set to ONE, shall mean that the file is an alias file and a pathname (3/13.4) shall be recorded as the contents of the file. If the Alias bit is ONE, the Version bit shall be ONE, and the Directory bit shall be ZERO.
7	Multi-Extent: If set to ZERO, shall mean that the file consists of only one file section and there shall be no “MF” extended attribute for the file. If set to ONE, shall mean that an “MF” extended attribute is recorded for the file.

15.1.6 File Unit Size (RBP 26)

If the value of this field is 0, the file is recorded in non-interleaved mode (see 3/13.5.1.1.2) and the Interleave Gap Size field shall be set to 0.

If the value of this field is not 0, the file is recorded in interleaved mode (see 3/13.5.1.1.1). The value of this field shall be the assigned file unit size for all file sections of the file, and there shall be no “FC” extended attribute recorded for the file.

15.1.7 Interleave Gap Size (RBP 27)

If the value of this field is 0, the file is recorded in non-interleaved mode and the File Unit Size field shall be set to 0.

If the value of this field is not 0, the file is recorded in interleaved mode. The value of this field shall be the assigned interleave gap size for all file sections of the file, and there shall be no “FC” extended attribute recorded for the file.

15.1.8 Volume Sequence Number (RBP 28)

If the value of this field is 0, an “MF” extended attribute shall be recorded for the file.

If the value of this field is not 0, there shall be no “MF” extended attribute recorded for the file and this field shall specify the assigned volume sequence number of the volume in the volume set on which the only file section of the file is recorded.

15.1.9 Length of File Identifier (=L_FI) (RBP 32)

This field shall specify the length, in bytes, of the File Identifier field of the Directory Record.

15.1.10 File Identifier (RBP 33)

This field shall specify an identification for the file. Except as specified in 3/9.5.3, 3/13.1.1.2 and 3/13.3.3, the contents of this field shall be d-characters (see 1/7.2).

15.1.11 Extended Attribute Area (RBP [L_FI+33])

This field shall contain an extended attribute area (see 3/14.1).

The first 4 bytes of this field shall be referred to as the Standard Defined Extended Attribute Existence field, shall be recorded as a `Uint32` (see 1/7.1.5) and shall specify the extended attributes recorded in the Extended Attributes Area field. A bit in the Standard Defined Extended Attribute Existence field corresponds to a particular extended attribute as shown in table 3/25. If the bit is ZERO, then that extended attribute shall not be recorded. If the bit is ONE, then that extended attribute shall be recorded.

Table 25 - Standard Defined Extended Attribute Existence field interpretation

Bit	Interpretation
0	Reserved: shall be set to ZERO.
1	“CS” extended attribute (3/15.3.2)
2	“PD” extended attribute (3/15.3.3)
3	“PM” extended attribute (3/15.3.4)
4	“RF” extended attribute (3/15.3.5)
5	“TF” extended attribute (3/15.3.6)
6	“TI” extended attribute (3/15.3.7)
7	“FC” extended attribute (3/15.3.8)
8	“CE” extended attribute (3/15.3.9)
9	“MF” extended attribute (3/15.3.10)
10	“UX” extended attribute (3/15.3.11)
11	“XR” extended attribute (3/15.3.12)
12-29	Reserved: Shall be set to ZERO.
30	“SU” extended attribute (3/15.3.13)
31	“AU” extended attribute (3/15.3.14)

15.1.12 Padding (RBP [L_XAA+L_FI+33])

This field shall be *rem(L_XAA+L_FI+1, 2)* bytes long and all bytes shall be set to #00.

15.1.13 File Version Number (RBP [*+L_XAA+L_FI+33])

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 32 768 to 65 535 inclusive are reserved for future standardisation.

15.2 Path Table Record

A Path Table Record shall be recorded in the format shown in table 3/26.

Table 26 - Path Table Record

RBP	Length	Name	Contents
0	2	Length of Path Table Record (=L_PTR)	`Uint16` (1/7.1.3)
2	8	Location of Directory	`Uint32BOTH` (3/8.6)
10	8	Data Length	`Uint32BOTH` (3/8.6)
18	7	Recording Date and Time	`shorttimestamp` (1/7.3.2)
25	1	File Flags	`Uint8` (1/7.1.1)
26	2	Reserved	#00 bytes
28	4	Parent Directory Number	`Uint16BOTH` (3/8.3)
32	1	Length of File Identifier (=L_FI)	`Uint8` (1/7.1.1)
33	L_FI	File Identifier	bytes
[L_FI+33]	L_XAA	Extended Attribute Area	bytes
[L_XAA+L_FI+33]	*	Padding	bytes

15.2.1 Length of Path Table Record (=L_PTR) (RBP 0)

This field shall specify the length, in bytes, of the Path Table Record.

15.2.2 Location of Directory (RBP 2)

This field shall specify the logical block number of the location of the extent in which the directory is recorded.

If the value of the Data Length field is 0, this field shall be set to 0, and the Version bit of the File Flags field shall be set to ONE.

15.2.3 Data Length (RBP 10)

This field shall specify the length of the directory (see 3/13.2.1).

If the value of this field is 0, the Location of Extent field shall be set to 0, and the Version bit of the File Flags field shall be set to ONE.

15.2.4 Recording Date and Time (RBP 18)

This field shall indicate the date and time of the day when this Path Table Record is recorded.

15.2.5 File Flags (RBP 25)

This field shall specify certain characteristics of the file as shown in table 3/24, with the following restrictions:

- The Directory bit shall be set to ONE.
- The Associated File, Record, Alias and Multi-Extent bits shall be set to ZERO.
- If the Version bit is set to ZERO, it shall mean that the directory shall contain only Directory Records with the Version bit set to ZERO. If the Version bit is set to ONE, it shall mean that the directory shall contain only Directory Records with the Version bit set to ONE.

15.2.6 Reserved (RBP 26)

This field shall be set to all #00 bytes.

15.2.7 Parent Directory Number (RBP 28)

This field shall specify the directory number (see 3/13.3) in the Path Table for the parent directory of the directory.

15.2.8 Length of File Identifier (=L_FI) (RBP 32)

This field shall specify the length, in bytes, of the File Identifier field of the Path Table Record.

15.2.9 File Identifier (RBP 33)

The field shall specify an identification of the directory. Except as specified in 3/9.5.4, 3/13.3 and 3/13.3.4, the contents of this field shall be d-characters (see 1/7.2).

15.2.10 Extended Attribute Area (RBP [L_FI+33])

15.2.11 Padding (RBP [L_XAA+L_FI+33])

This field shall be *rem(L_XAA+L_FI+1, 2)* bytes long. If this field is not of zero length, it shall be set to #00.

15.3 Extended Attributes

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

15.3.1 Extended Attribute format

An extended attribute shall be recorded as a short form extended attribute (see 3/15.3.1.1) if the length, in bytes, of the extended attribute is less than 255. An extended attribute shall be recorded as a long form extended attribute (see 3/15.3.1.2) if the length, in bytes, of the extended attribute is greater than or equal to 255.

15.3.1.1 Generic short form Extended Attribute format

The short form extended attribute shall be recorded in the format shown in table 3/27. The specification for each extended attribute shall specify the interpretation of the Attribute Subtype and Attribute Data fields of the extended attribute.

Table 27 - Generic short form Extended Attribute format

RBP	Length	Name	Contents
0	4	Attribute Type	`Uint32` (1/7.1.5)
4	1	Attribute Subtype	`Uint8` (1/7.1.1)
5	1	Attribute Length (=A_L)	`Uint8` (1/7.1.1)
6	A_L-6	Attribute Data	bytes

15.3.1.1.1 Attribute Type (RBP 0)

This field shall specify the type of the extended attribute.

15.3.1.1.2 Attribute Subtype (RBP 4)

This field shall specify the subtype of the extended attribute.

15.3.1.1.3 Attribute Length (=A_L) (RBP 5)

This field shall specify the length, in bytes, of the entire extended attribute, and shall be less than 255.

NOTE

It is recommended that the extended attribute length be an integral multiple of 2.

15.3.1.1.4 Attribute Data (RBP 6)

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

NOTE

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-6. 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.

15.3.1.2 Generic long form Extended Attribute format

The long form extended attribute shall be recorded in the format shown in table 3/28. The specification for each extended attribute shall specify the interpretation of the Attribute Subtype and Attribute Data fields of the extended attribute.

Table 28 - Generic long form Extended Attribute format

RBP	Length	Name	Contents
0	4	Attribute Type	`Uint32` (1/7.1.5)
4	1	Attribute Subtype	`Uint8` (1/7.1.1)
5	1	Attribute Length	`Uint8` (1/7.1.1) = 255
6	8	Actual Length (=A_L)	`Uint32BOTH` (3/8.6)
14	A_L-14	Attribute Data	bytes

15.3.1.2.1 Attribute Type (RBP 0)

This field shall specify the type of the extended attribute.

15.3.1.2.2 Attribute Subtype (RBP 4)

This field shall specify the subtype of the extended attribute.

15.3.1.2.3 Attribute Length (RBP 5)

This field shall specify 255.

15.3.1.2.4 Actual Length (=A_L) (RBP 6)

This field shall specify the length, in bytes, of the entire extended attribute and shall be greater than, or equal to, 255.

NOTE

It is recommended that the extended attribute length be an integral multiple of 2.

15.3.1.2.5 Attribute Data (RBP 14)

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

NOTE

The only meaning for the Actual 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-14. It is recommended that extended attributes with variable-sized data record the data length immediately after the Actual 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.

15.3.2 "CS" Extended Attribute

The "CS" extended attribute may be used to specify the coded character sets used in interpreting the contents of the current file.

The "CS" extended attribute shall be recorded as specified in 3/15.3.2.1 or 3/15.3.2.2.

15.3.2.1 Short "CS" Extended Attribute format

A short form "CS" extended attribute shall be recorded in the format shown in table 3/29.

Table 29 - Short CS Extended Attribute format

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	1	Attribute Length	`Uint8` (1/7.1.1)
6	8	Escape Sequences Length (=ES_L)	`Uint32BOTH` (3/8.6)
14	1	Character Set Type	`Uint8` (1/7.1.1)
15	ES_L	Escape Sequences	bytes

15.3.2.1.1 Attribute Type (RBP 0)

This field shall specify 1.

15.3.2.1.2 Attribute Subtype (RBP 4)

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

15.3.2.1.3 Attribute Length (RBP 5)

This field shall specify the length, in bytes, of the entire extended attribute.

15.3.2.1.4 Escape Sequences Length (=ES_L) (RBP 6)

This field shall specify the length, in bytes, of the Escape Sequence field.

15.3.2.1.5 Character Set Type (RBP 14)

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 Sequence field.

15.3.2.1.6 Escape Sequences (RBP 15)

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 contents 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.

15.3.2.2 Long "CS" Extended Attribute format

A long form "CS" extended attribute shall be recorded in the format shown in table 3/30.

Table30 - Long CS Extended Attribute format

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	1	Attribute Length	`Uint8` (1/7.1.1) = 255
6	8	Actual Length	`Uint32BOTH` (3/8.6)
14	8	Escape Sequences Length (=ES_L)	`Uint32BOTH` (3/8.6)
22	1	Character Set Type	`Uint8` (1/7.1.1)
23	ES_L	Escape Sequences	bytes

15.3.2.2.1 Attribute Type (RBP 0)

This field shall specify 1.

15.3.2.2.2 Attribute Subtype (RBP 4)

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

15.3.2.2.3 Attribute Length (RBP 5)

This field shall specify 255.

15.3.2.2.4 Actual Length (RBP 6)

This field shall specify the length, in bytes, of the entire extended attribute.

15.3.2.2.5 Escape Sequences Length (=ES_L) (RBP 14)

This field shall specify the length, in bytes, of the Escape Sequence field.

15.3.2.2.6 Character Set Type (RBP 22)

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 Sequence field.

15.3.2.2.7 Escape Sequences (RBP 23)

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 contents 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.

15.3.3 "PD" Extended attribute

The "PD" extended attribute may be used to align the first byte of the next extended attribute recorded in one or more of the extended attribute areas specified in 3/14.1 and 3/14.2.

The "PD" extended attribute shall be recorded as specified in 3/15.3.3.1 or 3/15.3.3.2.

15.3.3.1 Short "PD" Extended Attribute format

A short form "PD" extended attribute shall be recorded in the format shown in table 3/31.

Table 31 - Short PD Extended Attribute format

RBP	Length	Name	Contents
0	4	Attribute Type	`Uint32` (1/7.1.5) = 2
4	1	Attribute Subtype	`Uint8` (1/7.1.1) = 1
5	1	Attribute Length (=A_L)	`Uint8` (1/7.1.1)
6	A_L-6	Padding	bytes

15.3.3.1.1 Attribute Type (RBP 0)

This field shall specify 2.

15.3.3.1.2 Attribute Subtype (RBP 4)

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

15.3.3.1.3 Attribute Length (=A_L) (RBP 5)

This field shall specify the length, in bytes, of the entire extended attribute.

15.3.3.1.4 Padding (RBP 6)

The contents of this field are not specified by this Part.

15.3.3.2 Long "PD" Extended Attribute format

A long form "PD" extended attribute shall be recorded in the format shown in table 3/32.

Table 32 - Long PD Extended Attribute format

RBP	Length	Name	Contents
0	4	Attribute Type	`Uint32` (1/7.1.5) = 2
4	1	Attribute Subtype	`Uint8` (1/7.1.1) = 1
5	1	Attribute Length	`Uint8` (1/7.1.1) = 255
6	8	Actual Length (=A_L)	`Uint32BOTH` (3/8.6)
14	A_L-14	Padding	bytes

15.3.3.2.1 Attribute Type (RBP 0)

This field shall specify 2.

15.3.3.2.2 Attribute Subtype (RBP 4)

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

15.3.3.2.3 Attribute Length (RBP 5)

This field shall specify 255.

15.3.3.2.4 Actual Length (=A_L) (RBP 6)

This field shall specify the length, in bytes, of the entire extended attribute.

15.3.3.2.5 Padding (RBP 14)

The contents of this field are not specified by this Part.

15.3.4 “PM” Extended Attribute

The “PM” extended attribute specifies fields that can be used to support the file access permission scheme of ECMA-35 for the current file. It shall be recorded in the format shown in table 3/33.

NOTE

The “PM” extended attribute is an extension of the permissions field in ECMA-35 to allow for the case of writing information. In addition, it eliminates the inconsistencies of the specification in ECMA-35.

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.

Table 33 - PM Extended Attribute format

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	1	Attribute Length	`Uint8` (1/7.1.1) = 16
6	4	Owner Identification	`Uint16BOTH` (3/8.3)
10	4	Group Identification	`Uint16BOTH` (3/8.3)
14	2	Permissions	`Uint16` (1/7.1.3)

15.3.4.1 Attribute Type (RBP 0)

This field shall specify 3.

15.3.4.2 Attribute Subtype (RBP 4)

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

15.3.4.3 Attribute Length (RBP 5)

This field shall specify 16.

15.3.4.4 Owner Identification (RBP 6)

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.

15.3.4.5 Group Identification (RBP 10)

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 identification specified for the file. In this case, the Owner Identification field shall be set to 0.

15.3.4.6 Permissions (RBP 14)

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 table 3/34.

Table 34 - Allowed access

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.

NOTE

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 this Part. However, if a system uses the "PM" 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 fields of the "PM" extended attribute be mapped to the appropriate fields in the implementation.

15.3.5 "RF" Extended Attribute

The "RF" extended attribute specifies the record format and record attributes of the information in the current file. It shall be recorded in the format shown in table 3/35.

Table 35 - RF Extended Attribute format

RBP	Length	Name	Contents
0	4	Attribute Type	`Uint32` (1/7.1.5) = 4
4	1	Attribute Subtype	`Uint8` (1/7.1.1) = 1
5	1	Attribute Length	`Uint8` (1/7.1.1) = 16
6	1	Record Format	`Uint8` (1/7.1.1)
7	1	Record Display Attribute	`Uint8` (1/7.1.1)
8	8	Record Length	`Uint32BOTH` (3/8.6)

15.3.5.1 Attribute Type (RBP 0)

This field shall specify 4.

15.3.5.2 Attribute Subtype (RBP 4)

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

15.3.5.3 Attribute Length (RBP 5)

This field shall specify 16.

15.3.5.4 Record Format (RBP 6)

This field shall specify a number identifying the format of the information in the file as shown in table 3/36.

Table 36 - Information format

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.

This field shall contain 0 if one or both of the Directory bit and the Alias bit of the file flags field of the Directory Record for the file is set to ONE.

15.3.5.5 Record Display Attribute (RBP 7)

This field shall specify certain intended display attributes of the records in the file as shown in table 3/37.

Table 37 - Record Display Attribute interpretation

Number	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.

15.3.5.6 Record Length (RBP 8)

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.

15.3.6 "TF" Extended Attribute

The "TF" extended attribute specifies certain dates and times for the current file and shall be recorded in the format shown in table 3/38.

Table 38 - TF Extended Attribute format

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	1	Attribute Length (=A_L)	`Uint8` (1/7.1.1)
6	2	File Time Existence	`Uint16` (1/7.1.3
8	A_L-8	File Times	bytes

15.3.6.1 Attribute Type (RBP 0)

This field shall specify 5.

15.3.6.2 Attribute Subtype (RBP 4)

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

15.3.6.3 Attribute Length (=A_L) (RBP 5)

This field shall specify the length, in bytes, of the entire extended attribute.

15.3.6.4 File Time Existence (RBP 6)

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 table 3/39. 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 table 3/39 are reserved for future standardisation and shall be set to ZERO.

Table 39 - File Time Existence interpretation

Bit	Timestamp
0	File Creation Date and Time: the date and time of the day at which the file was created.
1	File Modification Date and Time: 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, if any.
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.
4	File Last Access Date and Time: the most recent date and time of the day of file creation or read access to the file prior to recording the "TF" extended attribute. This date and time shall not be earlier than the File Creation Date and Time, if any. *NOTE* *This departs a little from the interpretation in ISO/IEC 9945-1 in that read accesses since this extended attribute was recorded are ignored. This is intended to reduce updates on write-once media.*
5	File Last Backup Date and Time: the date and time of the day at which the file was last backed up.

15.3.6.5 File Times (RBP 8)

The dates and times specified in the File Times Existence field shall be recorded contiguously in this field, each as a `timestamp` (see 1/7.3.1), in ascending order of their bit positions.

15.3.7 "TI" Extended Attribute

The "TI" extended attribute specifies certain dates and times for the information in the current file and shall be recorded in the format shown in table 3/40.

Table 40 - TI Extended Attribute format

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	1	Attribute Length (=A_L)	`Uint8` (1/7.1.1)
6	2	Information Time Existence	`Uint16` (1/7.1.3)
8	D_L	Information Times	bytes

15.3.7.1 Attribute Type (RBP 0)

This field shall specify 6.

15.3.7.2 Attribute Subtype (RBP 4)

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

15.3.7.3 Attribute Length (=A_L) (RBP 5)

This field shall specify the length, in bytes, of the entire extended attribute.

15.3.7.4 Information Time Existence (RBP 6)

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 table 3/41. 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 table 3/41 are reserved for future standardisation and shall be set to ZERO.

Table41 - Information Time Existence interpretation

Bit	Timestamp
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.

15.3.7.5 Information Times (RBP 8)

The dates and times specified in the Information Times Existence field shall be recorded contiguously in this field, each as a `timestamp` (see 1/7.3.1), in ascending order of their bit positions.

15.3.8 "FC" Extended Attribute

If the "FC" extended attribute is recorded (see 3/13.5.1) for the current file, the following fields in the Directory Record (see 3/15.1) identifying this extended attribute shall have the following values:

– The Location of File Section field shall be set to all #00 bytes

– The Data Length field shall specify the length, in bytes, of the file.

– The Volume Sequence Number field shall contain the volume sequence number of the volume on which the Directory Record is recorded.

– The Directory and Multi-Extent bits in the File Flags field shall be set to ZERO

– The File Unit Size field (see 3/15.1.6) and Interleave Gap Size field (see 3/15.1.7) shall be set to 0.

The "FC" extended attribute shall be recorded as specified in 3/15.3.8.1 or 3/15.3.8.2.

15.3.8.1 Short "FC" Extended Attribute format

A short form "FC" extended attribute shall be recorded in the format shown in table 3/42.

Table 42 - Short FC Extended Attribute format

RBP	Length	Name	Contents
0	4	Attribute Type	`Uint32` (1/7.1.5) = 7
4	1	Attribute Subtype	`Uint8` (1/7.1.1) = 1
5	1	Attribute Length (=A_L)	`Uint8` (1/7.1.1)
6	A_L-6	File Contents	bytes

15.3.8.1.1 Attribute Type (RBP 0)

This field shall specify 7.

15.3.8.1.2 Attribute Subtype (RBP 4)

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

15.3.8.1.3 Attribute Length (=A_L) (RBP 5)

This field shall specify the length, in bytes, of the entire extended attribute.

15.3.8.1.4 File Contents (RBP 6)

The file shall be recorded in this field.

15.3.8.2 Long "FC" Extended Attribute format

A long form "FC" extended attribute shall be recorded in the format shown in table 3/43.

Table 43 - Long FC Extended Attribute format

RBP	Length	Name	Contents
0	4	Attribute Type	`Uint32` (1/7.1.5) = 7
4	1	Attribute Subtype	`Uint8` (1/7.1.1) = 1
5	1	Attribute Length	`Uint8` (1/7.1.1) = 255
6	8	Actual Length (=A_L)	`Uint32BOTH` (3/8.6)
14	A_L-14	File Contents	bytes

15.3.8.2.1 Attribute Type (RBP 0)

This field shall specify 7.

15.3.8.2.2 Attribute Subtype (RBP 4)

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

15.3.8.2.3 Attribute Length (RBP 5)

This field shall specify 255.

15.3.8.2.4 Actual Length (=A_L) (RBP 6)

This field shall specify the length, in bytes, of the entire extended attribute.

15.3.8.2.5 File Contents (RBP 14)

The file shall be recorded in this field.

15.3.9 "CE" Extended Attribute

The "CE" extended attribute shall be recorded in the format shown in table 3/44.

Table 44 - CE Extended Attribute format

RBP	Length	Name	Contents
0	4	Attribute Type	`Uint32` (1/7.1.5) = 8
4	1	Attribute Subtype	`Uint8` (1/7.1.1) = 1
5	1	Attribute Length	`Uint8` (1/7.1.1) = 70
6	8	Extent Location	`Uint32BOTH` (3/8.6)
14	8	Extent Size	`Uint32BOTH` (3/8.6)
22	8	Offset of the Additional Standard Attribute Area	`Uint32BOTH` (3/8.6)
30	8	Size of the Additional Standard Attribute Area	`Uint32BOTH` (3/8.6)
38	8	Offset of the Implementation Use Area	`Uint32BOTH` (3/8.6)
46	8	Size of the Implementation Use Area	`Uint32BOTH` (3/8.6)
54	8	Offset of the Application Use Area	`Uint32BOTH` (3/8.6)
62	8	Size of the Application Use Area	`Uint32BOTH` (3/8.6)

15.3.9.1 Attribute Type (RBP 0)

This field shall specify 8.

15.3.9.2 Attribute Subtype (RBP 4)

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

15.3.9.3 Attribute Length (RBP 5)

This field shall specify 70.

15.3.9.4 Extent Location (RBP 6)

This field shall specify the logical block number of the first logical block of the extent in which an Additional Extended Attribute Area (see 3/14.1.1 and 3/14.2.1) is recorded.

15.3.9.5 Extent Size (RBP 14)

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

15.3.9.6 Offset of the Additional Standard Attribute Area (RBP 22)

This field shall specify the relative byte offset of the first byte of an area recorded in the extent identified by the Extent Location field.

If the Additional Extended Attribute Area is recorded according to 3/14.2.1, this field shall be set to 0 and shall be ignored.

If the Additional Extended Attribute Area is recorded according to 3/14.1.1, this field shall specify the first byte of the Additional Standard Attribute Area (see 3/14.1.1).

15.3.9.7 Size of the Additional Standard Attribute Area (RBP 30)

This field shall specify the size, in bytes, of the Additional Standard Attribute Area of the Additional Extended Attribute Area identified by the Extent Location field.

If the Additional Extended Attribute Area is recorded according to 3/14.2.1, this field shall be set to 0 and shall be ignored.

If the Offset of the Additional Standard Attribute Area field is 0, the Size of the Additional Standard Attribute field shall be set to 0.

15.3.9.8 Offset of the Implementation Use Area (RBP 38)

This field shall specify the relative byte offset of the first byte of an area recorded in the extent identified by the Extent Location field.

If the Additional Extended Attribute Area is recorded according to 3/14.2.1, this field shall be set to 0 and shall be ignored.

If the Additional Extended Attribute Area is recorded according to 3/14.1.1, this field shall specify the first byte of the Implementation Use Area (see 3/14.1.1) in the Additional Extended Attribute Area.

15.3.9.9 Size of the Implementation Use Area (RBP 46)

This field shall specify the size in bytes of the Implementation Use Area of the Additional Extended Attribute Area identified by the Extent Location field.

If the Additional Extended Attribute Area is recorded according to 3/14.2.1, this field shall be set to 0 and shall be ignored.

If the Additional Extended Attribute Area is recorded according to 3/14.1.1, this field shall specify the size in bytes of the area identified by the Offset of the Implementation Use Area field.

15.3.9.10 Offset of the Application Use Area (RBP 54)

This field shall specify the relative byte offset of the first byte of an area recorded in the extent identified by the Extent Location field.

If the Additional Extended Attribute Area is recorded according to 3/14.2.1, this field shall be set to 0 and shall be ignored.

If the Additional Extended Attribute Area is recorded according to 3/14.1.1, this field shall specify the first byte of the Application Use Area (see 3/14.1.1) in the Additional Extended Attribute Area.

15.3.9.11 Size of the Application Use Area (RBP 62)

This field shall specify the size in bytes of the Application Use Area of the Additional Extended Attribute Area identified by the Extent Location field.

If the Additional Extended Attribute Area is recorded according to 3/14.2.1, this field shall be set to 0 and shall be ignored.

If the Additional Extended Attribute Area is recorded according to 3/14.1.1, this field shall specify the size in bytes of the area identified by the Offset of the Application Use Area field.

15.3.10 "MF" Extended Attribute

The "MF" extended attribute shall be used to identify all the file sections of the current file, except as specified in 3/13.1.1 and 3/13.5.1.

If an "MF" extended attribute is present, the following fields in the Directory Record identifying this extended attribute shall have the following values:

– the Location of File Section (see 3/15.1.2) and the Volume Sequence Number fields (see 3/15.1.8) shall be set to 0.

– the Data Length field (see 3/15.1.3) shall specify the sum of the lengths, in bytes, of the file sections described by the "MF" extended attribute.

The "MF" extended attribute shall be recorded as specified in 3/15.3.10.1 or 3/15.3.10.2.

15.3.10.1 Short "MF" Extended Attribute format

A short form "MF" extended attribute shall be recorded in the format shown in table 3/45.

Table 45 - Short MF Extended Attribute format

RBP	Length	Name	Contents
0	4	Attribute Type	`Uint32` (1/7.1.5) = 9
4	1	Attribute Subtype	`Uint8` (1/7.1.1) = 1
5	1	Attribute Length (=A_L)	`Uint8` (1/7.1.1)
6	A_L-6	File Sections	bytes

15.3.10.1.1 Attribute Type (RBP 0)

This field shall specify 9.

15.3.10.1.2 Attribute Subtype (RBP 4)

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

15.3.10.1.3 Attribute Length (=A_L) (RBP 5)

This field shall specify the length, in bytes, of the entire extended attribute.

15.3.10.1.4 File Sections (RBP 6)

This field shall contain a sequence of File Section Records (see 3/15.5) recorded contiguously, starting at the first byte of this field and ending at the last byte of this field. The number of File Section Records recorded in this field shall be (A_L-6)/20.

15.3.10.2 Long "MF" Extended Attribute format

A long form "MF" extended attribute shall be recorded in the format shown in table 3/46.

Table 46 - Long MF Extended Attribute format

RBP	Length	Name	Contents
0	4	Attribute Type	`Uint32` (1/7.1.5) = 9
4	1	Attribute Subtype	`Uint8` (1/7.1.1) = 1
5	1	Attribute Length	`Uint8` (1/7.1.1) = 255
6	8	Actual Length (=A_L)	`Uint32BOTH` (3/8.6)
14	A_L-14	File Sections	bytes

15.3.10.2.1 Attribute Type (RBP 0)

This field shall specify 9.

15.3.10.2.2 Attribute Subtype (RBP 4)

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

15.3.10.2.3 Attribute Length (RBP 5)

This field shall specify 255.

15.3.10.2.4 Actual Length (=A_L) (RBP 6)

This field shall specify the length, in bytes, of the entire extended attribute.

15.3.10.2.5 File Sections (RBP 14)

This field shall contain a sequence of File Section Records (see 3/15.5) recorded contiguously, starting at the first byte of this field and ending at the last byte of this field. The number of File Section Records recorded in this field shall be (A_L-14)/20.

15.3.11 "UX" Extended Attribute

The "UX" extended attribute shall specify certain information for the current file as specified in ISO/IEC 9945-1.

The "UX" extended attribute shall be recorded in the format shown in table 3/47.

Table 47 - UX Extended Attribute format

RBP	Length	Name	Contents
0	4	Attribute Type	`Uint32` (1/7.1.5) = 10
4	1	Attribute Subtype	`Uint8` (1/7.1.1) = 1
5	1	Attribute Length	`Uint8` (1/7.1.1)
6	2	Permissions	`Uint16` (1/7.1.3)
8	4	File Type	`Uint16BOTH` (3/8.3)
12	8	File Links	`Uint32BOTH` (3/8.6)
20	8	Uid	`Uint32BOTH` (3/8.6)
28	8	Gid	`Uint32BOTH` (3/8.6)
36	8	Major Device Identification	`Uint32BOTH` (3/8.6)
44	8	Minor Device Identification	`Uint32BOTH` (3/8.6)

15.3.11.1 Attribute Type (RBP 0)

This field shall specify 10.

15.3.11.2 Attribute Subtype (RBP 4)

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

15.3.11.3 Attribute Length (RBP 5)

This field shall specify the length of the “UX” extended attribute.

If the number in this field is 36, the length of the “UX” extended attribute shall be 36 and the Major Device Identification and Minor Device Identification fields shall not be present. Otherwise, the number in this field shall be 52, the length of the “UX” extended attribute shall be 52, and the Major Device Identification and Minor Device Identification fields shall be present.

15.3.11.4 Permissions (RBP 6)

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

– If the user's user ID is the same as the Uid field, then bits 6-11 shall apply.

– Otherwise, if the user's group ID is the same as the Gid field, then bits 3-5 and 9-11 shall apply.

– Otherwise, bits 0-2 and 9-11 shall apply.

The allowed access for the file is shown in table 3/48.

Table 48 - File Mode interpretation

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	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.
4	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.
5	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.
6	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.
7	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.
8	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.
9	Setuid: This bit shall be interpreted as the `S_ISUID` bit as specified in ISO/IEC 9945-1.
10	Setgid: This bit shall be interpreted as the `S_ISGID` bit as specified in ISO/IEC 9945-1.
11	Sticky: This bit shall be interpreted as the `C_ISVTX` bit as specified in ISO/IEC 9945-1.
12-15	Reserved: Shall be set to ZERO.

NOTE

File access schemes are subject to agreement between the originator and the 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 this Part. 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.*

15.3.11.5 File Type (RBP 8)

This field shall specify the type of the file as shown in table 3/49.

Table49 - File Type interpretation

Number	Interpretation
0	Shall mean that the interpretation of the file is not specified by this field.
1	Shall mean that the file shall be interpreted as a sequence of bytes, each of which may be randomly accessed
2	Shall mean that the file is a block special device file as specified by ISO/IEC 9945-1
3	Shall mean that the file is a character special device file as specified by ISO/IEC 9945-1
4	Shall mean that the file is a FIFO file as specified by ISO/IEC 9945-1
5	Shall mean that the file shall be interpreted according to the `C_ISSOCK` file type identified by ISO/IEC 9945-1
6-65 535	Reserved for future standardisation

If any of bits 1, 2, 3, and 6 of the File Flags field in the Directory Record identifying this extended attribute is ONE, or if this extended attribute is associated with a Path Table Record, this field shall be set to 0. If the number in this field is 0 or 1, the interpretation of the contents of the file shall be subject to agreement between the originator and recipient of the medium

15.3.11.6 File Links (RBP 12)

This field may be used to specify the number of files described by the directory hierarchy in which the current file is described that identify all of the same file sections constituting the current file. This number in this field shall be 1 if no such files are specified.

15.3.11.7 Uid (RBP 20)

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

NOTE

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.

15.3.11.8 Gid (RBP 28)

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

15.3.11.9 Major Device Identification (RBP 36)

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.

15.3.11.10 Minor Device Identification (RBP 44)

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.

15.3.12 “XR” Extended Attribute

The “XR” extended attribute identifies a previously recorded Directory Record.

NOTE

The “XR” extended attribute is intended to identify:

- *An ECMA-35 format Directory Record that may have some attributes recorded in an associated ECMA-35 Extended Attribute Record that are not identified by the Directory Record associated with the "XR" extended attribute.*
- *An earlier recorded Directory Record for the file that may have different values for and/or a different set of extended attributes than the Directory Record associated with the "XR" extended attribute.*

15.3.12.1 Interpretation of ECMA-35 Extended Attribute Records

If the "XR" extended attribute identifies a Directory Record that has the Version bit of the File Flags field (see 3/15.1.5) set to ONE, the identified Directory Record shall be interpreted according to 3/15.1.

If the "XR" extended attribute identifies a Directory Record that has the Version bit of the File Flags field (see 3/15.1.5) set to ZERO, the identified Directory Record shall be interpreted according to 9.1 of ECMA-35. If the identified Directory Record specifies that an Extended Attribute Record (see 9.5 of ECMA-35) is recorded for the file, the fields of the Extended Attribute Record shall be interpreted as specified in 9.5 of ECMA-35 except as follows:

- The Owner Identification field shall be interpreted according to3/15.3.4.4.
- The Group Identification field shall be interpreted according to 3/15.3.4.5.
- The Permissions field shall be interpreted according to 3/15.3.4.6.

NOTE

This assures a consistent interpretation of these three fields for both types of directories that may be included in a Part 3 directory hierarchy.

An "XR" extended attribute shall be recorded in the format shown in table 3/50.

Table 50 - XR Extended Attribute format

RBP	Length	Name	Contents
0	4	Attribute Type	`Uint32` 1/7.1.5) = 11
4	1	Attribute Subtype	`Uint8` (1/7.1.1) = 1
5	1	Attribute Length	`Uint8` (1/7.1.1) = 32
6	1	Flag	`Uint8` (1/7.1.1)
7	1	Reserved	#00 byte
8	4	Directory Record Length	`Uint16BOTH` (3/8.3)
12	4	Volume Sequence Number	`Uint16BOTH` (3/8.3)
16	8	Location of the Directory Record	`Uint32BOTH` (3/8.6)
24	8	Offset of the Directory Record	`Uint32BOTH` (3/8.6)

15.3.12.2 Attribute Type (RBP 0)

This field shall specify 11.

15.3.12.3 Attribute Subtype (RBP 4)

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

15.3.12.4 Attribute Length (RBP 5)

This field shall specify 32.

15.3.12.5 Flag (RBP 6)

This field shall be interpreted as shown in table 3/51.

Table 51 - Flag interpretation

Bit	Interpretation
0	Attribute: If set to ZERO, shall mean that the Directory Record identifying the “XR” extended attribute identifies all of the attributes identified by the Directory Record identified by the Offset of the Directory Record field in the “XR” extended attribute. If set to ONE, shall mean that the Directory Record identifying the “XR” extended attribute need not identify all such attributes.
1-7	Reserved for future standardisation and shall be set to ZERO.

15.3.12.6 Reserved (RBP 7)

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

15.3.12.7 Directory Record Length (RBP 8)

This field shall specify the length, in bytes, of the Directory Record identified by the Location of Directory Record field.

15.3.12.8 Volume Sequence Number (RBP 12)

This field shall specify the volume sequence number of the volume on which the Directory Record identified by the Location of Directory Record field is recorded.

15.3.12.9 Location of the Directory Record (RBP 16)

This field shall specify the logical block number of the logical block of the volume identified by the Volume Sequence Number field in which the first byte of the identified Directory Record is recorded.

15.3.12.10 Offset of the Directory Record (RBP 24)

This field shall specify the start of the identified Directory Record as a byte offset from the start of the logical block identified by the Location of Directory Record field.

15.3.13 “SU” Extended Attribute

The “SU” extended attribute shall be recorded as specified in 3/15.3.13.1 or 3/15.3.13.2.

15.3.13.1 Short “SU” Extended Attribute format

A short form “SU” extended attribute shall be recorded in the format shown in table 3/52.

Table 52 - Short SU Extended Attribute format

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	1	Attribute Length (=A_L)	`Uint8` (1/7.1.1)
6	32	Implementation Identifier	`regid` (1/7.4)
38	A_L-38	Implementation Use	bytes

15.3.13.1.1 Attribute Type (RBP 0)

This field shall specify 2 048.

15.3.13.1.2 Attribute Subtype (RBP 4)

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

15.3.13.1.3 Attribute Length (=A_L) (RBP 5)

This field shall specify the length, in bytes, of the entire extended attribute.

15.3.13.1.4 Implementation Identifier (RBP 6)

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` (1/7.4) includes the contents of the descriptors that specify the contents and attributes of the current file.

15.3.13.1.5 Implementation Use (RBP 38)

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

15.3.13.2 Long "SU" Extended Attribute format

A long form "SU" extended attribute shall be recorded in the format shown in table 3/53.

Table 53 - Long SU Extended Attribute format

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	1	Attribute Length	`Uint8` (1/7.1.1) = 255
6	8	Actual Length (=A_L)	`Uint32BOTH` (3/8.6)
14	32	Implementation Identifier	`regid` (1/7.4)
46	A_L-46	Implementation Use	bytes

15.3.13.2.1 Attribute Type (RBP 0)

This field shall specify 2 048.

15.3.13.2.2 Attribute Subtype (RBP 4)

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

15.3.13.2.3 Attribute Length (RBP 5)

This field shall specify 255.

15.3.13.2.4 Actual Length (=A_L) (RBP 6)

This field shall specify the length, in bytes, of the entire extended attribute.

15.3.13.2.5 Implementation Identifier (RBP 14)

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` (1/7.4) includes the contents of the descriptors that specify the contents and attributes of the current file.

15.3.13.2.6 Implementation Use (RBP 46)

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

15.3.14 "AU" Extended Attribute

The "AU" extended attribute shall be recorded as specified in 3/15.3.14.1 or 3/15.3.14.2.

15.3.14.1 Short "AU" Extended Attribute format

A short form "AU" extended attribute shall be recorded in the format shown in table 3/54.

Table 54 - Short AU Extended Attribute format

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	1	Attribute Length (=A_L)	`Uint8` (1/7.1.1)
6	32	Application Identifier	`regid` (1/7.4)
38	A_L-38	Application Use	bytes

15.3.14.1.1 Attribute Type (RBP 0)

This field shall specify 65 536.

15.3.14.1.2 Attribute Subtype (RBP 4)

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

15.3.14.1.3 Attribute Length (=A_L) (RBP 5)

This field shall specify the length, in bytes, of the entire extended attribute.

15.3.14.1.4 Application Identifier (RBP 6)

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` (1/7.4) includes the contents of the descriptors that specify the contents and attributes of the current file.

15.3.14.1.5 Application Use (RBP 38)

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

15.3.14.2 Long “AU” Extended Attribute format

A long form “AU” extended attribute shall be recorded in the format shown in table 3/55.

Table 55 - Long AU Extended Attribute format

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	1	Attribute Length	`Uint8` (1/7.1.1) = 255
6	8	Actual Length (=A_L)	`Uint32BOTH` (3/8.6)
14	32	Application Identifier	`regid` (1/7.4)
46	A_L-46	Application Use	bytes

15.3.14.2.1 Attribute Type (RBP 0)

This field shall specify 65 536.

15.3.14.2.2 Attribute Subtype (RBP 4)

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

15.3.14.2.3 Attribute Length (RBP 5)

This field shall specify 255.

15.3.14.2.4 Actual Length (=A_L) (RBP 6)

This field shall specify the length, in bytes, of the entire extended attribute.

15.3.14.2.5 Application Identifier (RBP 14)

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` (1/7.4) includes the contents of the descriptors that specify the contents and attributes of the current file.

15.3.14.2.6 Application Use (RBP 46)

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

15.4 Path Component

A Path Component shall be recorded in the format shown in table 3/56.

Table 56 - Path Component format

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 (1/7.1.2)

15.4.1 Component Type (RBP 0)

This field shall specify the component type as shown in table 3/57.

Table 57 - Component interpretation

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.

15.4.2 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.

15.4.3 Component File Version Number (RBP 2)

This field shall specify the file version number (see 3/13.7.2) 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 3/15.4.4) 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 3/15.4.4). The numbers 32 768 to 65 535 inclusive are reserved for future standardisation.

If the the entity identified by the Component Identifier field (see 3/15.4.4) is a directory, then the value of this field shall be 0.

NOTE

This allows versions of files and aliases to be specified in recorded pathnames.

15.4.4 Component Identifier (RBP 4)

This field shall identify the component.

15.5 File Section Record

A File Section Record shall be recorded as shown in table 3/58.

Table 58 - File Section Record format

RBP	Length	Name	Contents
0	4	Volume Sequence Number	`Uint16BOTH` (3/8.3)
4	8	Location of Extent	`Uint32BOTH` (3/8.6)
12	8	Data Length	`Uint32BOTH` (3/8.6)

15.5.1 Volume Sequence Number (RBP 0)

This field shall specify the ordinal number of the volume in the Volume Set on which the extent containing the file section identified by the File Section Record..

15.5.2 Location of Extent (RBP 4)

This field shall specify the logical block number of the logical block on the volume identified by the Volume Sequence Number field at which the first byte of the file section identified by the File Section Record.

If the value of this field is #FFFFFFFF, the value in the File Unit Size and Interleave Gap fields of the Directory Record identifying the “MF” extended attribute identifying this File Section Record shall be 0, the Data Length field in this File Section Record shall contain the size of the file section, and the file section shall be treated as if it consisted of all #00 bytes, and shall not be recorded.

15.5.3 Data Length (RBP 12)

This field shall specify the length, in bytes, of the file section identified by this File Section Record.

16 Levels of medium interchange

This Part specifies four nested 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 this Part.

16.1 Level 1

At level 1, the following restrictions shall apply:

- The number in any Length of File Identifier field and any Length of Component Identifier field shall not exceed 12.
- A File Identifier (see 3/13.7.1) for a directory shall conform to the schema shown in figure 3/11. A sequence of fileid-characters shall be a sequence of d-characters (see 1/7.2), excluding SPACE, COMMA, FULL STOP and REVERSE SOLIDUS characters except as part of a code extension character (see 1/7.2.9.1).

```
[Directory File Identifier]{
      <fileid-characters>1+8
}
```

Figure 11 - Level 1 directory file identifier schema

– A File Identifier for a file other than a directory shall conform to the schema shown in figure 3/12.

```
[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 12 - Level 1 nondirectory file identifier schema

– There shall not be more than Directory Record in a directory with the same contents in the File Identifier field.

– The length, in bytes, of a resolved pathname (see 3/13.4.1) shall not exceed 64.

– The length of a Directory Record shall not exceed 255 bytes.

– The length of a Path Table Record shall not exceed 255 bytes.

– Each file shall consist of only one file section.

– A volume set shall consist of one volume.

– The Alias and Multi-Extent bit in the File Flags field (see 3/15.1.5) of each Directory Record shall be ZERO.

– A file section shall not be recorded in interleaved mode (see 3/13.5.1.5.1).

– The depth of a directory hierarchy (see 3/13.1.4) shall not exceed 8.

NOTE

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

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.

16.2 Level 2

At level 2, the following restrictions shall apply:

- The number in any Length of File Identifier field and any Length of Component Identifier field shall not exceed 12.
- A File Identifier (see 3/13.7.1) for a directory shall conform to the schema shown in figure 3/13. A sequence of fileid-characters shall be a sequence of d-characters (see 1/7.2), excluding SPACE, COMMA, FULL STOP and REVERSE SOLIDUS characters except as part of a code extension character (see 1/7.2.9.1).

```
[Directory File Identifier for a directory]{
      <fileid-characters>1+8
}
```

Figure 13 - Level 2 directory file identifier schema

- A File Identifier for a file other than a directory shall conform to the schema shown in figure 3/14.

```
[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
      }
}
```

Figure14 - Level 2 nondirectory file identifier schema

- There shall not be more than Directory Record in a directory with the same contents in the File Identifier field.
- The length of a resolved pathname (see 3/13.4.1) shall not exceed 64.
- A volume set shall consist of one volume.
- The Alias bit in the File Flags field (see 3/15.1.5) of each Directory Record shall be ZERO.
- A file section shall not be recorded in interleaved mode (see 3/13.5.1.5.1).
- The depth of a directory hierarchy (see 3/13.1.4) shall not exceed 8.

NOTE

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

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.

16.3 Level 3

At level 3, the following restrictions shall apply:

- A volume set shall consist of one volume.
- A file section shall not be recorded in interleaved mode (see 3/31.5.1.5.1).

16.4 Level 4

At level 4, no restrictions shall apply.

Section 3 - Requirements for systems for volume and file structure

17 Requirements for the description of systems

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

NOTE

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 this Part.

18 Requirements for an originating system

18.1 General

The implementation shall be capable of recording a set of files, and all descriptors as specified in this Part, on a volume set according to one of the medium interchange levels specified in 3/16.

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 volume identification to use, then the implementation shall use the volume identification in the Primary Volume Descriptor of the prevailing Volume Descriptor of the prevailing Volume Descriptor Set of the volume.

If any information in the scope of a `regid` (see 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 3/18.2.3).

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.

18.2 Mandatory access by user

18.2.1 Files

The implementation shall obtain from the user the information that constitutes the set of files to be recorded.

18.2.2 File set

The implementation shall allow the user to specify which file set to use on a volume set and to identify the volumes on which the volume set is recorded.

If the user specifies a volume set without specifying which file set to use, then the implementation shall use the file set identified by the File Set Descriptor having File Set Descriptor Sequence Number 1 in the prevailing File System Descriptor Set of the volume set.

18.2.3 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.

Primary Volume Descriptor

- Descriptor Character Set
- Volume Identifier
- Volume Set Identifier
- Volume Set Size
- Volume Sequence Number
- Control Flags
- Maximum Interchange Level

- Maximum Character Set List

Supplementary Volume Descriptor

- Descriptor Character Set
- Volume Identifier
- Volume Set Identifier
- Supplementary Volume Descriptor Sequence Number
- Control Flags

File Set Descriptor

- File Set Character Set
- File Set Identifier
- File Set Descriptor Sequence Number
- Control Flags
- Maximum Interchange Level
- Maximum Character Set List

Directory Record

- Directory bit of the File Flags field
- File Identifier
- File Version Number

Path Table Record

- Directory bit of the File Flags field
- 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 Primary Volume Descriptor
- Maximum Character Set List field of a Primary Volume Descriptor
- Except as specified in 1/7.4 and 3/18.1, the Dirty and Protected bits of the Flags field of any `regid` (see 1/7.4)
- Existence bit of the File Flags field of a Directory Record or of a Path Table Record.

18.3 Optional access by user

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

18.3.1 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

- Implementation Identifier
- Logical Block Size
- End Transaction Descriptor Recording Rule
- Volume Set Creation Date and Time
- Descriptor Recording Date and Time
- Application Use

Supplementary Volume Descriptor

- Descriptor Recording Date and Time
- Application Use

File Set Descriptor

- Descriptor Character Set
- Domain Identifier
- File Set Creation Date and Time
- File Set Expiration Date and Time
- File Set Effective Date and Time
- Application Identifier
- Publisher Identifier
- Data Preparer Identifier
- Copyright File Identifier
- Abstract File Identifier
- Bibliographic File Identifier
- Application Use

Directory Record

- Directory bit of the File Flags field
- Existence bit of the File Flags field
- Associated File bit of the File Flags field
- Record bit of the File Flags field
- Protection bit of the File Flags field
- Alias bit of the File Flags field
- File Unit Size
- Interleave Gap Size
- Volume Sequence Number

Path Table Record

- Existence bit of the File Flags field
- Protection bit of the File Flags field
- Version bit of the File Flags field

Extended Attribute Descriptor

- Attribute Type
- Attribute Subtype
- Attribute Data

If the implementation allows the user to supply the information that is to be recorded in any of the following descriptor fields, the implementation shall not record the descriptor if the user does not supply the information.

Volume Partition Descriptor

- Descriptor Character Set
- Implementation Identifier
- Volume Partition Identifier
- Volume Partition Location
- Volume Partition Size
- Implementation Use

18.3.2 Records

If the implementation allows the user to specify that the information constituting a file is to be interpreted according to 3/15.3.5, the implementation shall obtain from the user the length of each record in the file and the bytes constituting the file.

18.3.3 File types

If the implementation allows the user to specify that a file is to be interpreted as (see 3/15.3.11.5) 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.

18.3.4 Permissions

The implementation should provide access to files and directories according to either, or both of, 3/15.3.4 or 3/15.3.11. However, as the implementation's security scheme might be incompatible with such an access scheme, the implementation is not required to provide such access.

18.4 Restrictions

18.4.1 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 $1 \le n < m$, 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 $1 \le n \le m$.

18.4.2 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 2 048.

18.4.3 File Times

If the “TF” extended attribute is not recorded for a file, then the implementation shall behave as if the “TF” extended attribute were recorded with the File Time Existence field having a value of 0.

18.4.4 Information Times

If the “TI” extended attribute is not recorded for a file, then the implementation shall behave as if the “TI” extended attribute were recorded with the Information Time Existence field having a value of 0.

18.4.5 Alternate Permissions

The implementation may ignore bits 0-3 of the Permissions field of the “PM” extended attribute (see 3/15.3.4).

If requested by the owner of the file, the implementation may ignore bits 4-7 of the “PM” extended attribute.

18.4.6 Extended Attributes

If the implementation cannot assure that a file would still conform to the specifications of an extended attribute, referred to as the current extended attribute, the implementation shall:

– not allow the user to modify the contents of the file, or;

– allow the user to modify the contents of the file and:

  - shall not record the current extended attribute with the recording of a Directory Record for the file, and;
  - shall include an “XR” extended attribute in each successor Directory Record for the file. The “XR” extended attribute shall identify a Directory Record of the file in which the current extended attribute is identified. The Attribute bit of the Flag field (see 3/15.3.12.5) of the “XR” extended attribute shall be set to ONE.

19 Requirements for a receiving system

19.1 General

The implementation shall be capable of reading the files, and the recorded descriptors as specified in this Part, from a volume set that has been recorded according to level 1 and one, or more, of levels 2, 3 and 4 as specified in 3/16.

The implementation shall be capable of interpreting 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 volume identification to use, then the implementation shall use the volume identification in the Primary Volume Descriptor of the prevailing Volume Descriptor of the prevailing Volume Descriptor Set of the volume.

The implementation shall allow the user to specify which file set to use on a volume set and to identify the volumes on which the volume set is recorded.

If the user specifies a volume set without specifying which file set to use, then the implementation shall use the file set identified by the File Set Descriptor having File Set Descriptor Sequence Number 1 in the prevailing File System Descriptor Set of the volume set.

19.2 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 3/15.3.5, the implementation shall make available to the user the length of each record in the file and the display attributes of the file.

19.2.1 File types

If the implementation allows the user to specify that a file is to be interpreted (see 3/15.3.11.5) as 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.

19.2.2 Permissions

The implementation should provide access to files and directories according to either, or both of, 3/15.3.4 or 3/15.3.11. However, as the implementation's security scheme might be incompatible with such an access scheme, the implementation is not required to provide such access.

19.3 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.

19.3.1 Descriptors

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

Primary Volume Descriptor

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

Supplementary Volume Descriptor

- Descriptor Character Set
- Volume Identifier
- Volume Set Identifier

File Set Descriptor

- Descriptor Character Set
- File Set Character Set
- File Set Identifier
- Maximum Interchange Level
- Maximum Character Set List
- Domain Identifier

Directory Record

- File Identifier
- File Version Number

Path Table Record

- Directory bit of the File Flags field
- File Identifier

19.4 Restrictions

19.4.1 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 2 048.

19.4.2 File Times

If the “TF” extended attribute is not recorded for a file, then the implementation shall behave as if the “TF” extended attribute were recorded with the File Time Existence field having a value of 0.

19.4.3 Information Times

If the “TI” extended attribute is not recorded for a file, then the implementation shall behave as if the “TI” extended attribute were recorded with the Information Time Existence field having a value of 0.

19.4.4 Alternate Permissions

The implementation may ignore bits 0-3 of the Permissions field of the “PM” extended attribute (see 3/15.3.4).

If requested by the owner of the file, the implementation may ignore bits 4-7 of the “PM” extended attribute.

Annex A

(normative)

Restrictions on a standard for recording

A standard for recording (see 1/5.13) used in conjunction with Part 3 shall have the following restrictions:

– The space on a medium shall be organised into containers, usually referred to as physical sectors. Each physical sector shall contain a data field, referred to as a sector (see 1/5.12) in this ECMA Standard.

– The size of a sector shall be at least 2 048 bytes.

The standard for recording shall specify:

– The means for specifying the address of the starting sector and the number of sectors in each track.

– The means for recording each track either incrementally according to 3/A.2 or track-at-once according to 3/A.3.

– The means to identify whether a track has been recorded incrementally according to 3/A.2 or track-at-once according to 3/A.3.

– The means for specifying the packet size (see 3/9.1.2.2.1) for a track recorded incrementally.

– The means for recording sectors with the attributes specified in 3/A.1.

– The means for identifying the attributes of each sector.

– A unique address for each sector.

– The length of each sector.

– The means for determining whether a sector is read-only or write-once.

– For media where sectors may only be recorded once, a means for detecting whether each sector has not yet been recorded.

– The means for identifying the starting track number in each session and number of tracks in each session.

A1 Sector attributes

A sector shall be assigned to one or more of the following attributes as shown in table 3/A.1.

Table A.1 - Sector attributes interpretation

Attribute Name	Interpretation
Mode 0	shall mean that the sector shall be interpreted as specified for Sector Mode (00) in ECMA-130.
Mode 1	shall mean that the sector shall be interpreted as specified for Sector Mode (01) in ECMA-130.
Mode 2	shall mean that the sector shall be interpreted as specified for Sector Mode (02) in ECMA-130.
Mode 2, Form 1	shall mean that the sector shall be interpreted as specified in 3/A.1.1.
Mode 2, Form 2	shall mean that the sector shall be interpreted as specified in 3/A.2.

A.1.1 Mode 2, Form 1

The first byte of the area (see 3/9.1.1.2) where information according to this ECMA Standard may be recorded shall be at an offset of *m* bytes, and the size of that area shall be 2 048 bytes. The value *m* shall be specified by the standard for recording.

A.1.2 Mode 2, Form 2

The first byte of the area (see 3/9.1.1.2) where information according to this ECMA Standard may be recorded shall be at an offset of *n* bytes, and the size of that area shall be greater than 2 048 bytes. The value *n* shall be specified by the standard for recording.

A.2 Incremental recording mode

If a track is assigned to be recorded in incremental mode (see 3/9.1.2.2.1), packets shall be recorded so that the sector numbers of the first sector in each packet of the track are in ascending order and that there shall be no unrecorded sectors prior to each packet.

A.3 Track-at-once recording mode

If a track is assigned to be recorded in track-at-once recording mode (see 3/9.1.2.2.1), all sectors of the track shall be recorded as a single unit.

Annex B

(informative)

Methods of interchange

Media may be interchanged between originating and receiving systems using different classes of implementations and media. The following classes of systems and media are illustrative, rather than exhaustive.

B.1 Classes of systems

An originating system or receiving system shall be classified into the following system classes according to the standards for recording and the volume and file structure standards to which the system conforms.

NOTE

Briefly, these system classes are:

S1 — Conforming to ECMA-130 and ECMA-35 with a CD-ROM reader.

S2 — Conforming to ECMA-130 and this Part with a CD-ROM reader.

S3 — Track-at-once CD-WO reader with support for Part 3.

S4 — Full annex A CD-WO reader with support for Part 3.

S5 — Track-at-once CD-WO reader/writer with support Part 3.

S6 — Full annex A CD-WO reader/Fixed packet CD-WO writer with support for Part 3.

S7 — Full annex A CD-WO reader/writer with support for Part 3.

There is no system class specified for a conforming ECMA-35 originating system as this Part does not provide for reading or recording all conforming ECMA-35 media.

B.1.1 System Class S1

A system shall belong to the system class S1 if it meets all of the following requirements:

- It is a conforming ECMA-35 receiving system.
- It is capable of reading media recorded according to ECMA-130.

B.1.2 System Class S2

A system shall belong to the system class S2 if it meets all of the following requirements:

- It is a conforming receiving system according to Part 3.
- It is capable of reading media recorded according to ECMA-130.

B.1.3 System Class S3

A system shall belong to the system class S3 if it meets all of the following requirements:

- It is a conforming receiving system according to Part 3.
- It is capable of reading media recorded according to annex A using the track-at-once recording method.

B.1.4 System Class S4

A system shall belong to the system class S4 if it meets all of the following requirements:

- It is a conforming receiving system according to Part 3.
- It is capable of reading media recorded according to annex A.

B.1.5 System Class S5

A system shall belong to the system class S5 if it meets all of the following requirements:

- It is a conforming receiving system and conforming originating system according to Part 3.
- It is capable of reading and writing media recorded according to annex A using the track-at-once recording method.

B.1.6 System Class S6

A system shall belong to the system class S6 if it meets all of the following requirements:

- It is a conforming receiving system and conforming originating system according to Part 3.
- It is capable of reading and writing media recorded according to annex A using the fixed-packet incremental recording method, and reading media recorded using the variable-length packet incremental recording method or track-at-once recording method.

B.1.7 System Class S7

A system shall belong to the system class S7 if it meets all of the following requirements:

- It is a conforming receiving system and conforming originating system according to Part 3.
- It is capable of reading and writing media recorded according to annex A.

B.2 Classes of media

A medium shall be classified into one of the following media classes according to the recording methods and the volume and file structure standard recorded on the medium.

NOTE

Briefly, these media classes are:

1R — ECMA-130 conforming media recorded according to Part 3 and ECMA-35, subject to constraints specified by Part 3.

2R — Part 3 recorded on ECMA-130 conforming media.

3U — Part 3 recorded on CD-WO media according to annex A using only the track-at-once method.

4U — Part 3 recorded on CD-WO media according to annex A.

B.2.1 Media Class 1R

A medium shall belong to the media class 1R if it meets all of the following requirements:

- The medium is recorded according to ECMA-130.
- It is recorded in one finalised session.

NOTE

A finalised session is a session marked, according to the standard for recording, as no longer being available for the recording of information.

- It contains only tracks recorded using the track-at-once method.
- A CD-ROM Volume Descriptor Set (see 2/8.3.1.1) is recorded in track number one of the volume.
- It is recorded according to one of the levels of medium interchange specified by Part 3 (see 3/16) and also to one of the levels of medium interchange specified by ECMA-35, subject to the additional restrictions of one of the levels of medium interchange specified by Part 3 and the restrictions in 3/13.1.1.

Media of the class 1R:

- May be read by systems of class S1 to S7.

- May be changed to media class 3U (see 3/B.2.3) by a system of class S5 or S7 by writing finalised sessions using the track-at-once recording methods.
- May be changed to media class 4U (see 3/B.2.4) by a system of class S6 or S7 using the fixed-length or variable-length packet incremental recording method.

B.2.2 Media Class 2R

A medium shall belong to the media class 2R if it meets all of the following requirements:

- The medium is recorded according to ECMA-130.
- It is recorded in one finalised session

NOTE

A finalised session is a session marked, according to the standard for recording, as no longer being available for the recording of information.

- It only contains tracks recorded using the track-at-once method
- It is recorded according to one of the levels of medium interchange specified by Part 3 (see 3/16).

Media of the class 2R:

- May be read by systems of class S2 to S7.
- May be changed to media class 3U (see 3/B.3.2) by a system of class S5 or S7 by writing finalised sessions using the track-at-once recording method.
- May be changed to media class 4U (see 3/B.3.4) by a system of class S6 or S7 using the fixed-length packet or variable-length packet incremental recording method.

B.2.3 Media Class 3U

A medium shall belong to the media class 3U if it meets all of the following requirements:

- The medium is recorded according to annex A.
- It is recorded in one or more finalised sessions

NOTE

A finalised session is a session marked, according to the standard for recording, as no longer being available for the recording of information.

- It only contains tracks recorded using the track-at-once method
- It is recorded according to one of the levels of medium interchange specified by Part 3 (see 3/16).

Media of the class 3U:

- May be read by systems of class S3 to S7.
- May be updated by a system of class S5 or S7 by writing finalised sessions using the track-at-once recording method.
- May be changed to media class 4U (see 3/B.2.4) by a system of class S6 or S7 using the fixed-length packet or variable-length packet incremental recording method, and by a system of class S5 or S7 using the track-at-once recording method.

B.2.4 Media Class 4U

A medium shall belong to the media class 4U if it meets all of the following requirements:

- The medium is recorded according to annex A.
- It is recorded only with one or more finalised sessions and, if any, one unfinalised session.

NOTE

A finalised session is a session marked, according to the standard for recording, as no longer being available for the recording of information.

- Each track may be recorded with any of the following recording methods: track-at-once or fixed-length packet incremental or variable-length packet incremental.
- It is recorded according to one of the levels of medium interchange specified by Part 3 (see 3/16).

Media of the class 4U:

- May be read by systems of class S4, S6, or S7.
- May be updated by a system of class S6 or S7 using the fixed-length packet or variable-length packet incremental recording method.

B.3 Media update transitions

Figure 3/B.1 depicts the paths of interchange allowed for media between systems. Boxes indicate media classes. A line from box x to box y is labelled by a list of system classes and means that media class x can be changed into media class y by systems of the designated system classes.

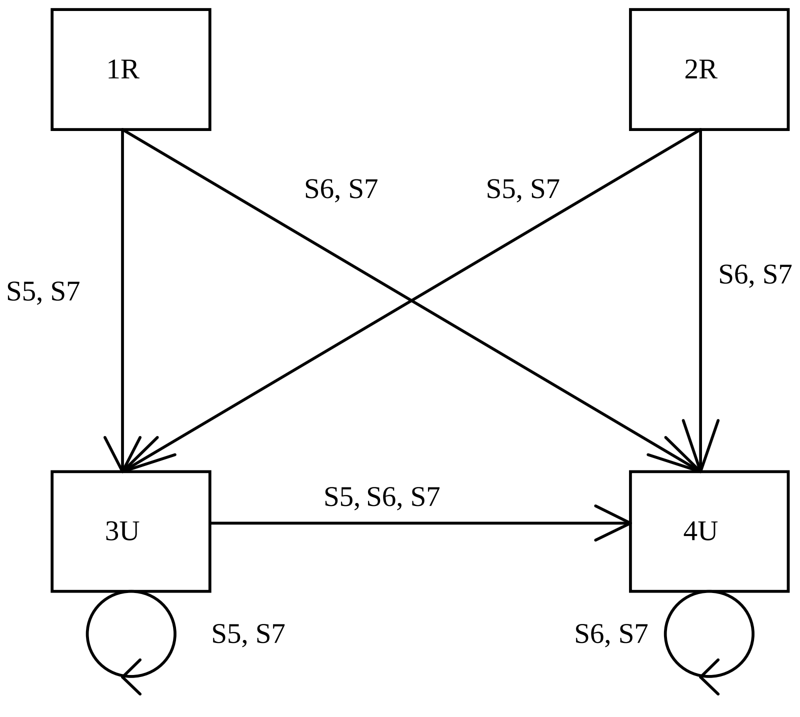


Figure B.1 - Media Update transition

Annex C

(informative)

CD-WO disk format and system requirements

C.1 CD-WO disk encoding standards

The encoding rules for Compact Disk Digital Audio tracks are as given in IEC 908 unless otherwise specified in this annex.

The encoding rules for data tracks are as given in ECMA-130 unless otherwise specified in this annex.

For the purposes of annex C, the addressable part of the medium that can be accessed independently of other addressable parts of the medium as specified in the standard for recording (see 1/5.13) shall be referred to as a sector. This differs from the use of the term sector (see 1/5.12) in the rest of this ECMA Standard as the Data field of a physical sector (see 3/C.2.1).

C.2 CD-WO disk format

C.2.1 Data sector organisation

Data sectors shall conform to Mode (00), Mode (01), or Mode (02) of ECMA-130.

If the value of the Pointer field of the q-Data field of q-Mode 1 in the Lead In Area (see ECMA-130) is #A0, and the value of the related P-Sec field (see ECMA-130) is either #20 or #10, a Mode (02) sector further contains a substructure in the User Data Area (see ECMA-130) to allow for high data integrity (Form 1 sector) and maximum data throughput (Form 2 sector).

NOTE

The q-Mode field in the q-channel (see ECMA-130) is not related to the Mode field in the Header field of a data sector.

In a Form 1 sector, the User Data Area is divided into a Subheader field (8 bytes), a Data field (2 048 bytes), an EDC field (4 bytes), and an ECC field (276 bytes).

Sector: 2 352 bytes							
Sync	**Header**		**Subheader**	**Data**	**EDC**	**ECC**	
	Sector address	**Mode**				**P-Parity**	**Q-Parity**
12 bytes	3 bytes	1 (02) bytes	8 bytes	2 048 bytes	4 bytes	172 bytes	104 bytes

Figure C.1 - Mode 2 Form 1 sector format

In a Form 2 sector, the User Data Area is divided into a Subheader field (8 bytes), a Data field (2 324 bytes) and 4 reserved bytes.

Sector: 2 352 bytes					
Sync	**Header**		**Subheader**	**Data**	**Reserved**
	Sector address	**Mode**			
12 bytes	3 bytes	1 (02) bytes	8 bytes	2 324 bytes	4 bytes

Figure C.2 - Mode 2 Form 2 sector format

The Subheader field of a Mode 2 Form 1 or Mode 2 Form 2 sector shall indicate whether the sector is a Mode 2 Form 1 or Mode 2 Form 2 sector.

C.2.2 Disk areas

With respect to data organisation, the Information Area (see ECMA-130) is divided into 5 parts:

– Power Calibration Area (PCA)

– Program Memory Area (PMA)

– Lead in Area

– Program Area

– Lead Out Area

C.2.2.1 Power Calibration Area

The Power Calibration Area is used for optimal write power control before actual recording.

C.2.2.2 Program Memory Area

The track numbers, and the begin and end addresses of already recorded tracks, are stored in the Program Memory Area, along with an optional disk identification. The PMA is updated each time the disk leaves a recorder.

After the last track has been recorded, complete track information from the PMA is stored in the Table of Contents (TOC) in the Lead In Area. This action is known as finalising the disk.

C.2.2.3 Lead in Area

The Lead In Area is reserved for recording the TOC according to IEC 908 or ECMA-130.

.0.0.1 Program Area

The Program Area is used for recording tracks containing user supplied information. These tracks can be both audio and data tracks.

When a disk leaves a recorder, whether finalised or not, all the recorded information must be contiguous.

NOTE

Compatibility of fixed/variable packet recording (see 3/C.2.3) with ECMA-130 is not guaranteed.

C.2.2.5 Lead Out Area

The Lead Out Area is recorded according to IEC 908 or ECMA-130. When finalising the disk, the Lead Out Area is recorded immediately after the last recorded track.

C.2.3 Recording

The CD-WO system gives the opportunity to record information “at-once” or “incrementally”. In uninterrupted recording, a complete disk is written at once (Disk-at-once recording). Incremental recording allows individual tracks or parts of tracks to be recorded in several separate recording actions at different times and on different recorders.

Incremental recording may comprise:

– Track at-once, in which each track is recorded in one uninterrupted stream. Within a track, compatibility with IEC 908 or ECMA-130 is achieved.

– Fixed Packet or Variable Packet, in which parts of a track are recorded in fixed or variable sized packets. The size of a packet is at least one User Data Sector.

C.2.4 Data linking rules

A recording shall start with one Link sector and four Run-in sectors, and shall end with two Run-out sectors. One set of Link, Run-in, User Data and Run-out sectors is called a packet.

The number of data sectors containing user-supplied information (User Data Sectors) in a Packet shall be the Packet Size. Bit 5 to bit 7 of the Mode field in the Header field of a data sector (see ECMA-130) shall contain an identification for the Link, Run-in, User Data and Run-out sectors as given in table 3/C.1.

Table C.1 - Data sector format

Bit: 7	6	5	Sector types
0	0	0	Data sector
0	0	1	Fourth Run-in sector
0	1	0	Third Run-in sector
0	1	1	Second Run-in sector
1	0	0	First Run-in sector
1	0	1	Link sector
1	1	0	Second Run-out sector
1	1	1	First Run-out sector

C.2.5 Data tracks

Every data track must start with a Pre gap, recording according to ECMA-130. Where no Pre gap is prescribed according to ECMA-130, a Pre gap of 2 seconds (150 sectors) must be recorded. Throughout the second part (see ECMA-130) of the Pre gap, the Track Descriptor sector (see 3/C.2.6) is recorded.

If a track is written incrementally, and if the first User Data sectors (see 3/C.2.4) are directly preceded by a Link sector and Run-in sectors, the Pre gap must end with the last Run-in sector, such that the header address of the first User Data sector in the track shall be the start address of the track.

If a track is assigned for Fixed Packet recording, all packets in that track shall have the same packet size.

If a track is assigned for Variable Packet recording, each packet in that track may have a different packet size.

C.2.6 Track Descriptor sector

A Track Descriptor sector contains a description of the recording method used in the corresponding track. The following information is encoded in a Track Descriptor sector:

– The size of the second part of the Pre gap

– The track number of the track

– The recording method used in the track, which is either Track at-once, Fixed Packet, or Variable Packet.

– The packet size if the recording method used in the track is Fixed Packet recording.

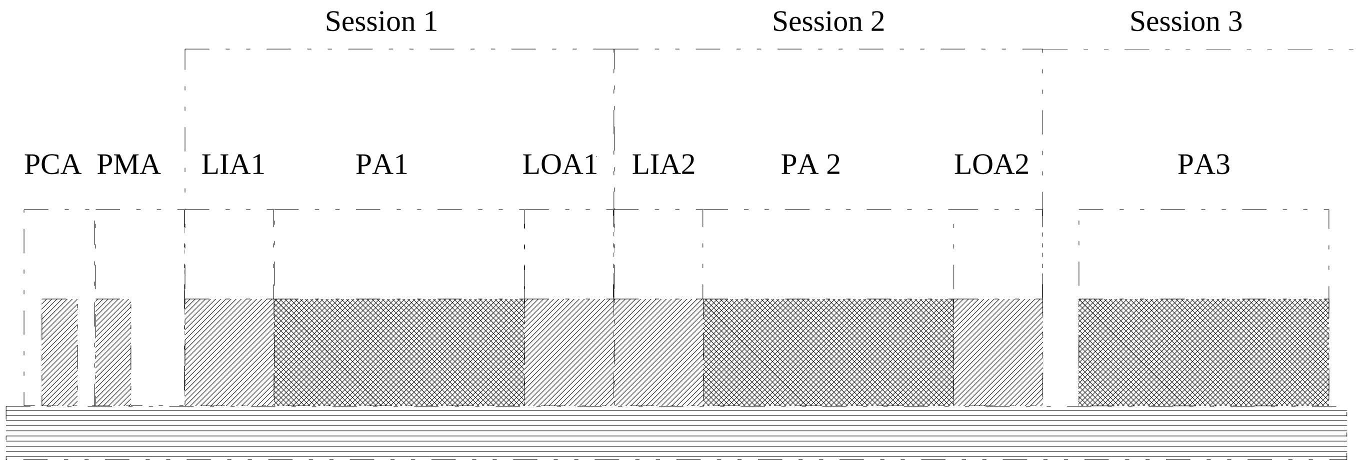
Optionally the description of previous tracks can be encoded in the Track Descriptor sector.

C.3 Multisession CD-WO disk format

A special version of the CD-WO disk is the multisession disk. This type of disk is specifically intended for data applications. A multisession disk comprises a Power Calibration Area (see 3/C.2.2.1), a Program Memory Area (see 3/C.2.2.2), and one or more sessions. Each session consists of a Lead-in area (see 3/C.3.1), a Program Area (see

3/C.3.2), and a Lead Out Area (see 3/C.3.3). As an option the first session(s) on a disk can be mastered (pre-mastered sessions).

Recorders and players conforming to ECMA-130 will in general be able to read the first session of a multisession disk, but not necessarily the later sessions.



Notation:

LIA = Lead in area, LOA = Lead out area, PA = Program area

Figure C.3 - Example of the layout of a multisession disk

C.3.1 Lead In Area

The Lead In Area of the first session must be recorded according to ECMA-130. The Lead In Area of the second or higher sessions must start at the end of the Lead Out area of the previous session.

In the Lead In Area, a TOC is recorded according to ECMA-130. In addition to the TOC, in the Lead In Area of a session, pointers are recorded in the Subcode Q-channel to the start address of the next possible session and to the greatest possible start position of the outermost Lead Out Area.

C.3.2 Program area

For the second and higher sessions on a disk, the start address of the Program Area of a session is encoded in the Lead In Area of the previous session. The start of the Program Area of the first session on a disk is recorded according to ECMA-130.

The first track number in a session is one higher than the last track number in the previous session. A maximum of 99 tracks can be recorded on 1 disk.

Before a disk leaves a recorder, all sessions except the last session must be finalised. A partially recorded Program Area may exist beyond the last finalised session.

C.3.3 Lead Out Area

The Lead Out Area of the first session of a disk must be recorded according to ECMA-130. The contents of a Lead Out Area are according to ECMA-130. In addition to the ECMA-130 specification, in the Lead Out Area of a session, pointers are recorded in the Subcode Q-channel to the start position of the first track in this session and to the greatest possible start position of the outermost Lead Out Area.

C.4 Addressing Methods

The addressing method expresses the relation between the Logical Sector numbering (see ECMA-130) and the sector Header Address. Two addressing methods are defined:

Method 1

Logical Sector Number *LSN = (((MIN*60) + SEC)*75 + FRAC) - 150* where *MIN*, *SEC* and *FRAC* is the Header address.

Method 2

*LSN*s up to and including the first User Data sector in a track are calculated by:

*LSN = (((MIN*60) + SEC)*75 + FRAC) - 150* where *MIN*, *SEC*, and *FRAC* is the Header address.

All following *LSN*s are calculated by counting all the User Data sectors in the track. Run-in sectors, Run-out sectors and link sectors are excluded.

Method 1 can be used over the entire disk. Method 2 is used only within an incrementally written track with fixed packets. Over the entire disk, the first User Data sector of each track has an addressing according to Method 1.

This means that between the end of an incrementally written track with fixed packets and the next track, there will be discontinuity in the addressing of the sectors.

STANDARD ECMA - 168

Volume and File Structure for Read-Only and Write-Once Compact Disk Media for Information Interchange

Part 4: Record Structure

Section 1 - General

1 Scope

Part 4 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 Part 4 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 Cross-reference

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.

- Data space of a file (see 4/6.1).
- If the records of the file are to be interpreted according to 4/9.2.4, 4/9.2.5, 4/9.2.6, 4/9.2.7 or 4/9.2.8 or are intended to be displayed according to 4/9.3, 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 4 specifies the following which may be used by other standards or Parts.

- Identification and specification of record types (see 4/9.2).
- Identification and specification of record display attributes (see 4/9.3).

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 4.

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 4.

8 Basic types

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

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

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 1/5.11) according to Part 4. 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 4/9.1).

9.1 Relationship to a file

Each MDU shall comprise a set of successive bytes of the data space of the file (see 4/6.1). 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 4 shall be one of the following types:

- padded fixed-length (4/9.2.1)
- fixed-length (4/9.2.2)
- variable-length-8 (4/9.2.3.1)
- variable-length-16 (4/9.2.3.2)
- variable-length-16-MSB (4/9.2.3.3)
- variable-length-32 (4/9.2.3.4)
- stream-print (4/9.2.4)
- stream-LF (4/9.2.5)
- stream-CR (4/9.2.6)
- stream-CRLF (4/9.2.7)
- stream-LFCR (4/9.2.8)

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 4/1.

```
[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 4/2.

```
[MDU ]{
      <record>
}
```

Figure 2 - Fixed-length record schema

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 (4/9.2.3.1)
– variable-length-16 (4/9.2.3.2)
– variable-length-16-MSB (4/9.2.3.3)
– variable-length-32 (4/9.2.3.4)

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 4/3, 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 block in which the RCW is recorded.
0 to $2^n - 2$	The RCW specifies the length of the record.

Figure 3 - RCW interpretation

NOTE

The length of the RCW is 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 4/4 where the RCW is recorded as an `Uint8` (1/7.1.1).

```
[MDU ]{
    <RCW>
    {
       <record>
    } 0+1
}
```

Figure 4 - Variable-length-8 record schema

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 4/6 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

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 4/7 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 4/8.

```
[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 4/9.

```
[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 4/10.

```
[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 4/11.

```
[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 4/12.

```
[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 4/9.2) specified in this Part, 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 this Part shall be assigned one of the following types of record display attributes

- LF-CR (4/9.3.1)
- first byte position (4/9.3.2)
- implied (4/9.3.3)

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

NOTE

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 this Part.

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 4/9.2, 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 4/9.3, 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 2 048.

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 4/9.2, 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 4/9.3, 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 2 048.