DEVELOPMENT STANDARD

QIC-113
Revision G
15 Jun 95

HOST INTERCHANGE FORMAT

Quarter-Inch Cartridge Drive Standards, Inc.

311 East Carrillo Street
Santa Barbara, California 93101
Telephone (805) 963-3853
Fax (805) 962-1541
www.qic.org

(See important notices on the following page)

Important Notices

This document is a development standard adopted by Quarter-Inch Cartridge Drive Standards, Inc. (QIC). This document may be revised several times during the development cycle. It is intended solely as a guide for companies interested in developing products which can be compatible with other products developed using this document. QIC makes no representation or warranty regarding this document, and any company using this document shall do so at its sole risk, including specifically the risks that a product developed will not be compatible with any other product or that any particular performance will not be achieved. QIC shall not be liable for any exemplary, incidental, proximate or consequential damages or expenses arising from the use of this document. This development standard defines only one approach to the product. Other approaches may be available in the industry.

This development standard is an authorized and approved publication of QIC. The underlying information and materials contained herein are the exclusive property of QIC but may be referred to and utilized by the general public for any legitimate purpose, particularly in the design and development of quarter-inch tape cartridge drive subsystems. This development standard may be copied in whole or in part *provided* that no revisions, alterations or changes of any kind are made to the materials contained herein. Only QIC has the right and authority to revise or change the material contained in this development standard, and any revisions by any party other than QIC are totally unauthorized and specifically prohibited.

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

Revision History

1) Revision D of the QIC-113 specification is a complete re-write of the specification and therefore includes changes on all pages. Support was added for SCSI tape drives, QIC-40/80 tape drives and the logical format is defined for QIC-3010/3020-MC tape drives within this specification. Additionally, support for non-DOS operating systems was added. Operating systems supported by the extended format include: Windows NT, OS/2, Novell 2.X, Novell 3.X, Novell 4.X, UNIX and DOS with long file names.

Revision E:

2) In Section 8 a paragraph was added that describes handling of the File Set Byte Offset in compressed volumes when the Directory Last Feature is added.

3) Clean-up of typos and clarifications including bit fields for Novell attributes as defined by Novell. Also in Section 4.0 the Minor Specification Revision Number was changed to reflect Revision E.

4) In Section 7.1.1.1.1.1 an addition was made to accommodate a Root Entry bit and its definition.

Revision F:

5) In Section 2.5, a Data Description ID was added for Windows 95.

6) In Sections 4 and 5, a Format and OS Type field value for Volume Table Entries was added for Windows 95.

7) In Sections 6.1.2, 6.2.1, 7, and 7.1.1.1, the File Set Data Entry signature was corrected to reflect the hex value 0x33CC33CC which when written to tape in little endian format would result in ‘CC33CC33’ as least significant bytes precede most significant bytes.

8) In Section 7.1.1.3.3.5 an XOS Record ID was added for Windows NT Registry.

9) Section 7.1.1.3.3.6 was added with an XOS Record ID specification for Windows 95 Registry.

10) In Section 7.2.0.1.1.1, the Root Entry discussion was enhanced to include root naming conventions as well as root traversal alternatives.

11) In Section 7.2.0.1.1.4, a Windows 95 Data Description Structure was added.

12) In Section 9, the terms First Block Offset, File Set Byte Offset, and Byte Count were changed to Next Extent Offset, Uncompressed Volume Byte Offset, and Frame Size respectively. Also, the term Compression Segment Structure is defined in terms of a Compression Extent and Compression Frames.

13) In Section 5, the requirement to write an all zeroes volume table segment for SCSI drives was removed.

14) In Sections 7 and 8, SCSI drives use a file mark to separate Data and Directory sections.

Revision G:

15) Section 5.2, addition of unicode tape name within the volume table.

16) Section 5.3, addition of volume table overflow extension.

1. SCOPE AND INTRODUCTION 1

2. CONVENTIONS 2

2.1. BIT FIELDS 2

2.2. DATE/TIME FIELDS 2

2.3. NUMERIC FIELDS 2

2.4. NAME FIELDS 3

2.5. UNDEFINED FIELDS 3

3. DEFINITIONS 3

3.1. COMPRESSION EXTENT 3

3.2. COMPRESSION FRAME 3

3.3. DATA AREA DATA 3

3.4. DATA AREA ID 3

3.5. DATA AREAS 3

3.6. DATA DESCRIPTION ENTRY 4

3.7. DATA DESCRIPTION ENTRY ID 4

3.8. DATA DESCRIPTION PORTION 4

3.9. DATA DESCRIPTION STRUCTURE 4

3.10. DATA ENTRY 4

3.11. DIRECTORY ENTRY 4

3.12. EXTENDED OS RECORD 5

3.13. EXTENDED OS RECORD DATA 5

3.14. EXTENDED OS RECORD ID 5

3.15. FILE SET 5

3.16. FILE SET DATA SECTION 5

3.17. FILE SET DIRECTORY SECTION 5

3.18. FIXED PORTION 5

3.19. LINK SUB-SECTION 5

3.20. LOGICAL SEGMENT 5

3.21. MEDIA ENDING OFFSET 5

3.22. MEDIA SEQUENCE NUMBER 5

3.23. NAME PORTION 6

3.24. OBJECT 6

3.25. PATH ENTRY 6

3.26. SEGMENT GAP 6

3.27. SIGNATURE 6

3.28. VENDOR SPECIFIC PORTION 6

3.29. VOLUME 6

3.30. VOLUME DATA AREA 6

4. HEADER FRAME *FOR SCSI DRIVES* 7

5. VOLUME TABLE SEGMENT *FOR SCSI DRIVES* 8

5.1. VOLUME TABLE ENTRY EXTENSION 10

5.2. VOLUME TABLE EXTENSION FOR UNICODE TAPE NAMES 10

5.3. VOLUME TABLE OVERFLOW EXTENSION 11

6. IMPLEMENTATION OF QIC-113 *FOR QIC-40/80 DRIVES* 12

7. VOLUME DATA AREA FOR BASIC DOS INTERCHANGE 14

7.1. FILE SET DIRECTORY SECTION 14
7.1.1. DIRECTORY SECTION POSITION 14
7.1.2. LINK SUB-SECTION 15
7.1.3. DIRECTORY ENTRY 15
7.1.4. DIRECTORY ENTRY ORDERING 16

7.2. FILE SET DATA SECTION 18
7.2.1. DATA ENTRY LAYOUT 19
7.2.2. PATH ENTRY LAYOUT 19

8. VOLUME DATA AREA FOR EXTENDED OPERATING SYSTEMS 20

8.1. FILE SET DATA SECTION 21

8.1.1. DATA ENTRY 21

8.2. FILE SET DIRECTORY SECTION 25
8.2.1. DIRECTORY ENTRY ORDERING 34
8.2.2. LINK SUB-SECTION 37

9. VOLUME COMPRESSION 38

9.1. COMPRESSION EXTENTS 38
9.1.1. NON-SEGMENT SPANNING COMPRESSION EXTENTS 38
9.1.2. SEGMENT SPANNING COMPRESSION EXTENTS 38
9.1.3. COMPRESSION EXTENT LAYOUT 38
9.1.4. COMPRESSION FRAME 39
9.1.5. EXAMPLE SEGMENT OF A NON-SEGMENT SPANNING VOLUME 40
9.1.6. EXAMPLE SEGMENTS OF A SEGMENT SPANNING VOLUME 40

1. SCOPE AND INTRODUCTION

The purpose of this document is to provide a logical format for interchange of data on tape. This document describes a complete tape layout for drives in the SCSI drive class and a tape volume format for drives in both the mini-cartridge class and SCSI drive class.
This document is organized in the following fashion:

SECTION 2 -	Conventions.
SECTION 3 -	Definitions.
SECTION 4 -	Format specification for a tape header frame so that this format can be identified for drives in the SCSI tape drive class.
SECTION 5 -	Volume table descriptions for SCSI tape drive class.
SECTION 6 -	Implementation method for QIC-40 and QIC-80 drives.
SECTIONS 7&8 -	File set logical format for application of this specification to drives of all tape classes. ***These are the only sections of this specification that apply to QIC-3010-MC and QIC-3020-MC.***
SECTION 9 -	Volume compression.

QIC-113 is required for implementation of the QIC-3010-MC and QIC-3020-MC specifications. The matrix below illustrates the elements of QIC-113 that must be complied with for a given operating system. The Basic DOS level of interchange is required for QIC-3010-MC and QIC-3020-MC compliance. If the non-DOS OS extensions are supported, specification compliance requires that the software system also has the ability to convert the non-DOS operating system information into the Basic DOS format to enhance the ability to interchange data.

Specification Feature	DOS	Novell	OS/2	Unix	NT
Basic DOS Read/Write	YES	YES	YES	YES	YES
Directory First	YES	YES	YES	YES	YES
Directory Last	YES	YES	YES	YES	YES
Compressed STAC -no spanning	YES	YES	YES	YES	YES
Compressed STAC -with spanning	YES	YES	YES	YES	YES
No compression	YES	YES	YES	YES	YES
Novell Extensions Read/Write		YES			
OS/2 Extensions Read/Write			YES		
Unix Extensions Read/Write				YES	
Windows NT Extensions Read/Write					YES

Specifications Requirements Matrix

2. CONVENTIONS

2.1. BIT FIELDS

Unless otherwise noted, the attribute specified is TRUE if the corresponding bit is set (=1).

For example:

bit 0: directory

object is a directory if bit 0 = 1.

Also, bit fields or structures returned by the operating system that do not have an explicit format defined by this specification are assumed to be stored as provided by the OS.

2.2. DATE/TIME FIELDS

Date fields (unless otherwise noted) are in the following form (sub-fields are little endian):
Invalid or undefined date/times should be set have all bytes set to 0xFF.

Offset,	***Length***	***Description***
0,	4	Date/Time in number of seconds since midnight of 1970 GMT Examples: January 1, 1970 00:00:00 GMT = 0 January 1, 1970 23:59:59 GMT = 86399 February 6, 2106 06:28:15 GMT = 0xFFFFFFFF
4,	4	Time Zone*/Microseconds tz=[-1440 through +1440 inclusive specifying minute variation from GMT] us=[# microsecs past second specified by Date/Time field, 0 if unavailable] field value = (*tz* left shifted 20 bits into most significant 12 bits) ORed with *us*

*This is provided for anticipated interchange with other formats that may specify a time zone.

Date/time fields explicitly denoted as *Short Date/Time* fields will have the following format:

Offset,	***Length***	***Description***
0,	4	bits 31-25: Year-1970 (0-127 = 1970-2097) bits 24-0: Month, day, hour, minute, second (sc+60*(mn+60*(hr+24*(dy+31*mo))) where sc=second (0-59), mn=minute (0-59), hr=hour (0-23), dy=day (0-30), mo=month (0-11)

2.3. NUMERIC FIELDS

Unless otherwise noted, numeric fields are stored little endian (least significant bytes precede most significant bytes).

Decimal digits will be used in any decimal notation for numeric fields.

Hexadecimal digits preceded by '0x' will be used in any hexadecimal notation for numeric fields.

2.4. NAME FIELDS

All name fields, unless otherwise noted, are stored in Unicode format where each byte pair contains a Unicode character set value.

2.5. UNDEFINED FIELDS

Unless otherwise noted, any field whose value is undefined, cannot be determined, or is reserved should be set to 0.

Exceptions:

- Unknown Data Entry size and Data Area size fields; every byte of these should be set to 0xFF when the actual value is unknown.
- Unknown Date/Time fields should have all bytes set to 0xFF.

3. DEFINITIONS

For the purposes of this Standard, the following definitions apply. All terms in bold within the definitions of this section also have entries in this section.

3.1. COMPRESSION EXTENT

A contiguous set of **Compression Frames** with a leading Uncompressed Volume Byte Offset to enable locating files within a compressed volume without decompressing preceding segments.

3.2. COMPRESSION FRAME

A unit of compressed data that may be decompressed without any previous or subsequent compression history.

3.3. DATA AREA DATA

The portion of a **Data Area** that contains **Extended OS Record**s or blob data bytes.

3.4. DATA AREA ID

Identifies a **Data Area**'s type and matches the **Data Description Entry ID** from the corresponding **Data Description Entry** found in the object's **Directory Entry**.

3.5. DATA AREAS

One or more areas within a **Data Entry** that contains data bytes or variable length attributes or augmentation data for a given object with respect to a particular file system. Each area contains a header with signature and ID.

3.6. DATA DESCRIPTION ENTRY

An entry within the **Data Description Portion** of a **Directory Entry** that contains information about an object with respect to a particular file system.

3.7. DATA DESCRIPTION ENTRY ID

The values in the following table are used for identifying the type and structure of a **Data Description Entry**. These values are also used in the *Native File System* field of the **Fixed Portion** of a **Directory Entry** as well as in **Path Entry** components to identify the name space of the component.

The priority of each ID is used to determine its ordering within the **Data Description Portion** of the **Directory Entry**. Note that the priorities may change in the future as new **Data Description ID**s are inserted; however, the relative priorities of the current items listed below will not change.

Each ID remains compatible with new revisions of OS as long as the information provided for file system elements remains consistent.

Value	***Priority***	***Description/Compatibility***
0	10, lowest	Vendor Specific
1	8	UNIX
2	9	DOS through 6.0
3	2	Novell 3.x
4	6	OS/2 through 2.1
5	4	Windows NT 1.0
6	7	AFP
7	0, highest	Data
8	3	Novell 2.x
9	1	Novell 4.x
10	5	Windows 95

3.8. DATA DESCRIPTION PORTION

The portion of a **Directory Entry** that contains information about an object with respect to specific file systems.

3.9. DATA DESCRIPTION STRUCTURE

The structure contained in a **Data Description Entry** defined by the information required for an object of a particular file system.

3.10. DATA ENTRY

A single object's entry within the **File Set Data Section.**

3.11. DIRECTORY ENTRY

A single object's entry within the **File Set Directory Section.**

3.12. EXTENDED OS RECORD

A variable length record contained in the **Data Area** corresponding to a **Data Description ID** that specifies a particular file system supporting extended attributes for an object.

3.13. EXTENDED OS RECORD DATA

Data (variable length) portion of an **Extended OS Record**.

3.14. EXTENDED OS RECORD ID

The ID that specifies the type of data contained in the **Extended OS Record Data**.

3.15. FILE SET

The entire collection of **objects** that are included in a **volume**.

3.16. FILE SET DATA SECTION

The initial sub-area of a **Volume Data Area** and contains the data associated with all objects contained within the **volume**.

3.17. FILE SET DIRECTORY SECTION

The sub-area that follows the **File Set Data Section**. This section contains basic object information and also defines the hierarchical relationship between the objects within the **volume**.

3.18. FIXED PORTION

The portion of a **Directory Entry** that is fixed in size and provides fields of information common to all objects contained within a **volume**.

3.19. LINK SUB-SECTION

This sub-area is appended to the **File Set Directory Section** when the **volume** spans more than one media. It contains information describing the amount of information contained on the current and previous media portions of the **volume**.

3.20. LOGICAL SEGMENT

A data block of 32K bytes (may or may not include ECC data).

3.21. MEDIA ENDING OFFSET

An offset that specifies the ending uncompressed byte offset of the **File Set Data Section** of a particular media.

3.22. MEDIA SEQUENCE NUMBER

The media sequence that specifies the ordering of a media within the overall **Volume Data Area** for **volume**s that span multiple media.

3.23. NAME PORTION

The portion of a **Directory Entry** in the basic format that specifies the entry's corresponding object's name.

3.24. OBJECT

A file, directory, device, or any other entity included in a **volume** and is represented by an entry in both the **File Set Data Section** and **File Set Directory Section**.

3.25. PATH ENTRY

An entry within a **Data Entry** that specifies an object's hierarchical position within the **volume**.

3.26. SEGMENT GAP

The area of physical media between the **File Set Data Section** and the **File Set Directory Section** as the latter is always segment boundary aligned.

3.27. SIGNATURE

A four byte field that provides for object field identification when scanning/resynching in the event that portions of a **volume** are corrupted.

3.28. VENDOR SPECIFIC PORTION

The portion of a **Directory Entry** in the basic format that specifies vendor specific data and whose layout is defined by the vendor that created the **volume**.

3.29. VOLUME

A volume consists of a **File Set Directory Section** and a **File Set Data Section** concatenated together. One or more such volumes may be combined on tape to fill the logical data area of the cartridge. If necessary, these sections may be repeated on multiple cartridges to complete the entire volume image.

3.30. VOLUME DATA AREA

The region of media occupied by all the data contained within a **volume**.

4. HEADER FRAME *FOR SCSI DRIVES*

The first data frame written on the tape shall contain information identifying the interchange format used. This header block shall be replicated five times (as frames 1-5 on the tape) to allow reliable identification of the format used.

If this header frame is not found repeated at least twice at the beginning of the tape, the tape does not conform to this interchange standard.

The contents of the header block shall be as follows:

Offset,	***Length***	***Description***
0,	12	Header frame signature. Must equal the ASCII uppercase string "HEADERQIC113"
12,	1	Revision Level encoded as a binary byte 7 = Revision G
13,	1	Interchange format: 0 = QIC 113 1 = vendor unique 2-255 reserved
14,	1	Quick File Access 0 = non-QFA tape 1 = QFA tape 2-255 reserved
15,	8	Reserved
23,	44	Tape name. Any ASCII string, left justified and space filled.
67,	4	Short Date/Time that the tape name was written.
71,	49	Vendor Unique - Identifier string.
120,	1	Two's complement sum of bytes 0-119
121,	391	Reserved (set to 0)

5. VOLUME TABLE SEGMENT *FOR SCSI DRIVES*

Each time a volume is added to a tape, a new volume entry is appended to the volume table which is replicated at EOD for non-QFA tapes and appended to the directory partition for QFA tapes. In this way, the final volume table segment, located by seeking to end of data or end of directory partition and backspacing to previous filemark, contains all volume entries for the tape.

For example (VDA = Volume Data Area, VTS = Volume Table Segment, and + indicates a filemark),

non-QFA tapes:

VDA + VTS(1 entry) + VDA + VTS(2 entries) + ... + VDA + VTS(n entries)

QFA directory partition:

VTS(1 entry) + VTS(2 entries) + ... + VTS(n entries)

The volume table shall contain an entry for each file set stored on the cartridge. Each entry within the table shall be 128 bytes long (unless augmented with an optional extended entry described in the next section).

The first 4 bytes of a used entry shall hold the uppercase ASCII string "VTBL" as a signature. *Following all used entries, the end of the volume table shall be indicated by an entry with an undefined signature*. If the first table slot contains no signature, the cartridge is empty. If the last slot contains a signature, the cartridge is full.

It shall be required that each successive volume table entry allocate a segment range beyond any currently in use. Thus the ending segment number of the final table entry acts as a starting point for further allocation.

The volume table entry for supplementary cartridges in a multi-cartridge file set shall be required to be identical to that of the initial cartridge, except for the following fields:

Offset(s)	
	4-7: Number of logical segments used.
	56: Multiple cartridge bit (bit 1).
	57: Multi-cartridge sequence number.
	66-83: Vendor extension data.
	96-103: File Set Data Section size (when directory last bit (bit 5, byte 56) is set)

All character strings shall not contain leading blanks (unless empty) and shall be blank padded. Blank - Hex 20. All other character codes are permitted, and system specific.

Each volume table entry shall be formatted as follows:

Offset,	*Length*	*Description*
0,	4	Volume entry signature. Must equal the ASCII uppercase string "VTBL" for all used entries.
4,	4	Number of logical segments used. Represents the total number of logical segments needed to store the data of a volume on a tape. If the volume spans multiple tapes, this number represents only the data on a particular tape.
8,	44	Volume entry description string. May be any string of ASCII characters and space filled. First byte must equal zero if none specified.
52,	4	Short Date/Time volume was created.
56,	1	Volume flags: bit 0: Vendor specific. Indicates that remainder of volume entry is vendor specific (only this bit and the previous bytes (0-55) are defined). bit 1: Multiple cartridge. Indicates that volume spans to another cartridge. bit 2: Non-verification. Indicates file set was written without verification. bit 3: Reserved (set to 0) bit 4: Compressed data segment spanning. bit 5: Directory last. Indicates that File Set Directory Section follows File Set Data Section. Always set if providing extended OS support. bit 6: Reserved (set to 0) bit 7: Reserved (set to 0)
57,	1	Multi-cartridge sequence number. Verifies a correct multiple cartridge loading sequence. Initial cartridge = 1, next cartridge = 2, etc. Can be greater than 1 only if the volume is the first volume table entry for the cartridge.
58,	2	Major Specification Revision Number = 113
60,	2	Minor Specification Revision Number = 7 (revision G)
62,	14	Vendor extension data. Reserved for unique vendor extensions to the volume entry. If used, should be combined with a signature to insure recognition of valid data only.
76,	4	Starting physical block address (QFA)
80,	4	Ending physical block address (QFA)
84,	8	File set password. First byte zero if none specified; otherwise, may be any ASCII string. Controls file set read access only.
92,	4	File Set Directory Section size. Size (in bytes) of area reserved for the File Set Directory Section on this cartridge. The directory table will not be required to fill this area completely.
96,	8	File Set Data Section size. Total size (in bytes) of the area reserved for the File Set Data Section. Includes all cartridges of multicartridge volumes unless the Directory Last bit (bit 5 of byte 56) is set in which case this number reflects the sum of the File Data Section sizes up through this cartridge only.
104,	2	OS version number. Associated with the OS Type field (e.g., if OS is Novell 3.12, OS Type field would = 4, and this field would contain 3 for major version, and 12 for minor version. byte 0: Major version number. byte 1: Minor version number.
106,	16	Source drive volume label. Any ASCII string, first byte zero if none specified.
122,	1	Logical device file set originated from.
123,	1	Reserved (set to 0)

124,	1	Compression method used. See QIC-123 for definition. bit 0-5: Compression code field (0x3F indicates vendor specific). bit 6: Set to 0. bit 7: Compression was used.
125,	1	Format & OS Type. Identifies whether volume data area is formatted in Basic DOS or Extended format. If extended format is used, also indicates operating system volume data originated from. The OS type is informational only, the important feature about this field is that if its value is NOT 1, extended format is used. 0 = Extended Format (originating OS unknown) 1 = DOS Basic Format (originating OS type not preserved) 2 = Unix 3 = OS/2 4 = Novell NetWare 5 = Windows NT 6 = DOS extended format (to differentiate between basic DOS format) 7 = Windows 95
126,	2	Reserved (set to 0)

5.1. VOLUME TABLE ENTRY EXTENSION

Volume table entries may be extended to support features such as unicode volume names and passwords by appending to them an extended entry. These extension entries will be identified by the ASCII signature "XTBL" and should be preserved even if the contents are ignored by software that does not support the extended information. Volume table extensions are required for extended OS support when unicode is used. A volume table extension is logically paired with the volume table entry that immediately precedes the volume table extension.

XTBL entries may also be implemented for QIC-40/80 volume entries when the volume entry complies to the QIC-113 revision D (or later) specification (see section 5).

Offset,	***Length***	***Description***
0,	4	Extended volume entry signature. Must equal the ASCII uppercase string "XTBL".
4,	88	Volume Name (Unicode)
92,	16	Volume Password (Unicode)
108,	20	Reserved (set to 0)

5.2. VOLUME TABLE EXTENSION FOR UNICODE TAPE NAMES

Unicode tape names may be implemented through the use of a Unicode Tape ID entry, “UTID”. For formats such as QIC-80 that accommodate overwrite, only one UTID entry may be made within the volume table. For formats that do not support overwrite, the last UTID entry encountered within the volume table area shall be used as the tape’s name.

Offset,	***Length***	***Description***
0,	4	Volume entry signature. Must equal the ASCII uppercase string “UTID”.
4,	88	Unicode Tape Name (left justified and space filled)
92,	36	Reserved (set to 0)

5.3. VOLUME TABLE OVERFLOW EXTENSION

For formats such as QIC-80 which has a finite volume table size, the volume table may be extended into another segment using an Extended Volume Table Entry, “EXVT”. This entry shall be used to allocate a previously unused tape segment for the purpose of extending the volume table segment. This 128 byte entry shall be the last entry within the segment containing this portion of the volume table.

Offset,	***Length***	***Description***
0,	4	Volume entry signature. Must equal the ASCII uppercase string “EXVT”.
4,	2	Parent Segment Number. Word: Number of segment containing this entry.
6,	2	Child Segment Number. Word: Number of next segment to extend the parent volume table (for this cartridge).
8,	119	Reserved (set to 0)

6. IMPLEMENTATION OF QIC-113 *FOR QIC-40/80 DRIVES*

The extended OS features of QIC-113 can be taken advantage of on QIC-40/80 drives. When a QIC-113 extended OS volume is to be writtten, bit 0 of byte 56 in the Volume Table Segment shall be set. This bit indicates that a Vendor Specific volume follows and therefore prevents old software from becoming confused by the new logical format. To identify the Vendor Specific volume as compliant with this specification, the major and minor revision numbers of this spec will be written as little endian binary words in the Vendor Extension Data at offsets 58 and 60 respectively. For example, QIC-113 rev. G would be encoded as follows:

offsets 58/59 = 113,
offsets 60/61 = 7

When the above criteria is met the host software shall interpret the remainder of the volume as adhering to the QIC-113 revision F specification. The remainder of the vendor specific area is defined by the vendor. The remainder of the volume entry follows the following format:

Offset,	***Length***	***Description***
84,	8	File set password. First byte zero if none specified; otherwise, may be any ASCII string. Controls file set read access only.
92,	4	File Set Directory Section size. Size (in bytes) of area reserved for the File Set Directory Section on this cartridge. The directory table will not be required to fill this area completely.
96,	8	File Set Data Section size. Total size (in bytes) of the area reserved for the File Set Data Section. Includes all cartridges of multicartridge volumes unless the Directory Last bit (bit 5 of byte 56) is set in which case this number reflects the sum of the File Data Section sizes up through this cartridge only.
104,	2	OS version number. Associated with the OS Type field (e.g., if OS is Novell 3.12, OS Type field would = 4, and this field would contain 3 for major version, and 12 for minor version. byte 0: Major version number. byte 1: Minor version number.
106,	16	Source drive volume label. Any ASCII string, first byte zero if none specified.
122,	1	Logical device file set originated from.
123,	1	Reserved (set to 0)
124,	1	Compression method used. See QIC-123 for definition. bit 0-5: Compression code field (0x3F indicates vendor specific). bit 6: Set to 0. bit 7: Compression was used.

125,	1	Format & OS Type. Identifies whether volume data area is formatted in Basic DOS or Extended format. If extended format is used, also indicates operating system volume data originated from. The OS type is informational only, the important feature about this field is that if its value is NOT 1, extended format is used. 0 = Extended Format (originating OS unknown) 1 = DOS Basic Format (originating OS type not preserved) 2 = Unix 3 = OS/2 4 = Novell NetWare 5 = Windows NT 6 = DOS extended format (to differentiate between basic DOS format) 7 = Windows 95
126,	2	Reserved (set to 0)

Note also that when a volume entry complies with QIC-113 revision D or later, it may also take advantage of the volume table extension entries (see section 5.1).

7. VOLUME DATA AREA FOR BASIC DOS INTERCHANGE

A volume consists of a File Set Directory Section and a File Set Data Section concatenated together. One or more such volumes may be concatenated on tape to fill the logical data area of the cartridge. If necessary, these sections may be repeated on multiple cartridges to complete the entire volume image.

If the Directory Last bit (byte 56, bit 5) of the volume's volume table entry is not set, the File Set Directory Section of the volume shall precede the File Set Data Section and shall be contained completely on the first cartridge used. If the Directory Last bit is set then the directory section will follow the data section and an image of the directory shall be written to the end of each tape in a linked tape set. A Link Sub-Section is appended as part of the directory section when this occurs. Note that the directory section is a partial directory on all but the last media. The last entry of these partial directories is determined by the last data entry started on that media.

Volume = File Set Directory Section + File Set Data Section

OR optionally,

Volume = File Set Data Section + Segment Gap + File Set Directory Section

or,

Volume (spanning multiple media) =
(File Set Data Section + Segment Gap + (File Set Directory Section + Link Sub-Section)) ...

Note also, that for SCSI drives a file mark always separates the File Set Data Section from the File Set Directory Section regardless of which order they are placed on the tape. This aids in the recovery of data; by simply traversing the tape by filemarks, one can easily find directory vs data sections.

7.1. FILE SET DIRECTORY SECTION

The File Set Directory Section is composed of one or more variable length Directory Entries concatenated together. This section is aligned on a segment boundary to allow it to be located easily during restore.

The case of an empty root directory yields a totally empty file set. This shall be recorded by storing a zero total data section size within the file set volume table entry.

7.1.1. DIRECTORY SECTION POSITION

Byte 56 of the Volume Table Entry contains the Volume Flags. Bit 5 is defined as the Directory Last Flag. The directory shall only reside at the start of the volume when the bit is not set and it shall only reside at the end of the volume when the bit is set. When the bit is set the File Set Directory shall be segment aligned and located by subtracting the Directory Section Size (offsets 92-95) rounded up to an integral number of segments from the Ending Segment Number (offsets 6-7). This is to accommodate finding the start of the Directory Section. The Directory Section Size shall include not only the size of the Directory Section but the size of Linked Tape Structure (for linked tapes only), described below, plus four (4) for the size of the Directory Section Ending Offset.

7.1.2. LINK SUB-SECTION

When the Directory Last Bit is set for linked volumes, the File Set Directory shall be written at the end of each tape in the linked set but shall be complete only to the point of the current tape. The last File Set Directory entry included represents the last file that had at least 1 byte of its signature 0x33CC33CC included in the data section. The last tape in the linked set shall have a complete image of the Directory Section.

To better enable locating files on linked tapes an additional data structure is required. This structure records the ending uncompressed byte offset of each tape in the linked set. The structure immediately follows the Directory Section. This structure is only used if the Directory Last Bit is set.

Offset,	***Length***	***Description***
0,	4	Directory Section Ending Offset. Points to the last byte in the File Set Directory Section for this tape.
4,	n	File Set Directory Section. Offset "n" is equal to the Directory Section Ending Offset.
4+n,	4	Linked Tape Table signature. Must equal the ASCII uppercase string "LTLT" and indicates the start of the Linked Tape Table Structure.
8+n,	1	Number of entries that follow. Indicates how many quadwords follow. Entries are ordered from the first tape in the linked set to the current tape.
9+n,	8	Ending uncompressed byte offset. Indicates the offset of the last data byte written to the corresponding tape in the chain.

7.1.3. DIRECTORY ENTRY

A variable length entry shall be made in the File Set Directory Section for each file and directory. These entries consist of three parts: a fixed portion, an optional vendor specific portion, and a name portion.

Directory Entry = (Fixed Portion) + [Vendor Specific Portion] + (Name Portion)

7.1.3.1. FIXED PORTION LAYOUT

Offset,	*Length*	*Description*
0,	1	Fixed & Vendor Portion size. Size of Fixed Portion (NOT including this byte) + Size of Vendor Specific Portion of entry.
1,	1	Attributes: (use default if source environment does not support particular attribute) bit 0: Read access allowed. (default - set) bit 1: Write access allowed. (default - set) bit 2: Execute access allowed. (default - set) bit 3: Hidden. (default - not set) bit 4: System. (default - not set) bit 5: Subdirectory bit 6: Last entry in current directory. bit 7: Last entry in entire directory table.
2,	4	Modification (Short) Date/Time.
6,	4	Data Entry size. If a file: Size of data header + count of the file's data bytes If a "linked" file OR device OR an empty directory: Size of data header If a non-empty directory: Zero See section 6.2 for a description of the "data header" component.
10,	1	Extra File Info: bit 0-5: File Info. 2=File unreadable at backup time. bit 6-7: Reserved (set to 0)

7.1.3.2. VENDOR SPECIFIC PORTION

Vendor specific data may follow the fixed portion when the size at offset 0 of the fixed portion exceeds 10. The layout of this area is defined by the vendor. A signature or some other sequence of bytes should be included in this area if used to insure proper identification.

7.1.3.3. NAME PORTION LAYOUT

Offset,	*Length*	*Description*
0,	1	Name size (not including this byte).
1,	n	Name. ASCII file or directory name string.

7.1.4. DIRECTORY ENTRY ORDERING

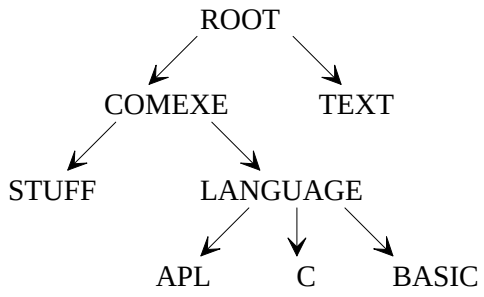
The ordering of the entries within the File Set Directory Section will be consistent with a breadth-first, preorder directory traversal of the source tree. A preorder traversal in this context is defined as visiting each entry in a given directory level prior to descending into the left-most (occurs first at parent level) child directory through right most (occurs last at parent level) child directory. Note that a node in this traversal scheme is considered to be an entire directory level (all entries) and not the individual entries.

There are two advantages to this method. First, the path alterations required to perform the entire traversal are minimized. Second, and more importantly, the data area of any complete sub-tree will be contiguous upon the tape, allowing rapid recovery of a directory and its children.

The root directory entries will appear first (in the order provided by the file system). Its last entry (as will the last entry of all directories) shall be indicated by the setting of the 'Last entry in directory' bit of the Attributes Byte. Following will be the first root directory entries, and below it, all of its sub-directory entries. The end of the entire directory table is indicated by the final entry having a special end-of-backup bit set.

Ordering example:

Given a source tree as follows:

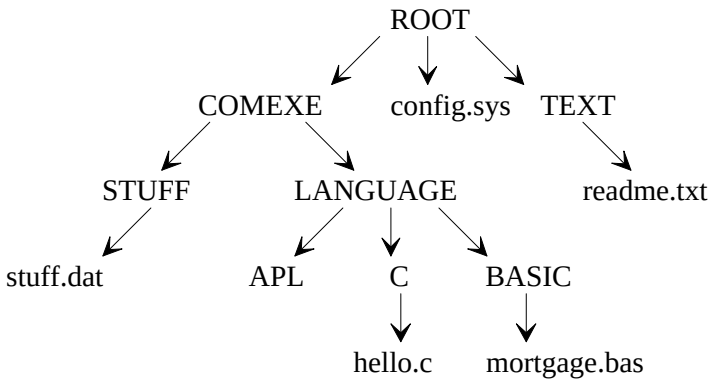


The directories would be traversed in the following order:

```
ROOT
        COMEXE
                STUFF
                LANGUAGE
                        APL
                        C
                        BASIC
        TEXT
```

Note that each directory node listed in the example above includes all of the entries at that level.

The following example shows the ordering of the individual entries (files are represented by lower case names):



Results in:

ENTRY	ATTRIBUTE BYTE Directory	Last entry in directory	Last entry in entire backup set
COMEXE	1	0	0
config.sys	0	0	0
TEXT	1	1	0
STUFF	1	0	0
LANGUAGE	1	1	0
stuff.dat	0	1	0
APL	1	0	0
C	1	0	0
BASIC	1	1	0
hello.c	0	1	0
mortgage.bas	0	1	0
readme.txt	0	1	1

7.2. FILE SET DATA SECTION

The data section of a file set consists of data bytes for all of the items listed in the directory table (except for non-empty directories), concatenated together. Data items occur in the data section in the same order as do their corresponding entries within the directory table.

Each block of data shall be preceded by a data header formatted as shown below. To achieve maximum density, no sector or segment boundaries shall be observed when this area is created.

Directory entries are included into this area only when they are for an empty directory. This inclusion is to allow as much of the tree structure as possible to be re-built if the directory table (or cartridge containing it) is lost. Such entries consist of a data header only, with no additional data following.

7.2.1.DATA ENTRY LAYOUT

Offset,	*Length*	*Description*
0,	4	Signature = 0x33CC33CC
4,	n	File Set Directory Entry. Copy of entry from the directory table for this item.
4+n,	m	Path Entry.
4+n+m,	k	File data (if present)

7.2.2.PATH ENTRY LAYOUT

Offset,	*Length*	*Description*
0,	1	Path size. Does not include this byte.
1,	n	Path. Full ASCII path to the directory this item resides in (items null separated).

8. VOLUME DATA AREA FOR EXTENDED OPERATING SYSTEMS

At the highest level, a volume consists of a File Set Data Section and a File Set Directory Section composed of Data Entries and Directory Entries respectively. These sections are separated by a Segment Gap as the File Set Directory Section is always segment boundary aligned on the physical media (to improve locating this section during volume restoration). Note, that when creating this type of volume, the Directory Last bit (byte 56, bit 5) of the volume's volume table entry must always be set.

Volume = File Set Data Section + Segment Gap + File Set Directory Section

These are repeated on each media if the volume is required to span more than one media. A Link Sub-Section is provided when this occurs and is appended to the File Set Directory Section on each media. Note the Directory Section is a partial directory on all but the last media. The last entry of these partial directories is determined by the last data entry started on that media.

Volume (spanning multiple media) =
(File Set Data Section + Segment Gap + (File Set Directory Section+Link Sub-Section)) ...

Note also, that for SCSI drives a file mark always separates the File Set Data Section from the File Set Directory Section regardless of which order they are placed on the tape. This aids in the recovery of data; by simply traversing the tape by file marks, one can easily find directory vs data sections.

A more complete breakdown of a Volume Data Area would appear as follows:
(Each indented item is fully contained in the preceding unindented level)

Volume Data
- File Set Data Section
  - Data Entries... (one or more)
    - Signature 0x33CC33CC
    - Directory Entry
      - Fixed Portion
      - Data Description Portion
        - Data Description Entries... (one or more)
    - Path Entry
    - Data Area...
      - Signature 0x66996699
      - Data Area ID
      - Data Area Data (Null | Blob | Multirecord)
- Segment Gap
- File Set Directory Section (segment aligned)
  - Directory Entries... (one or more)
  - Link Section (if volume spans multiple media)
    - Signature 'LTLT'
    - Media Sequence #
    - Media Ending Offsets...

8.1. FILE SET DATA SECTION

The File Set Data Section is the first section of a volume and consists of one or more File Set Data Entries concatenated together. These entries contain a copy of the File Set Directory Entry as well as any associated data bytes and extended OS data. The ordering is consistent with that of the Directory Section.

To achieve maximum density, no sector or segment boundaries shall be observed when this area is created.

All entries contained in the File Set Directory Section shall also be found in the File Set Data Section.

Note that some information in the copy of the Directory Entry found in this section may not be accurate (specifically Data Entry size and Data Area size) as it is not always possible to have complete information before the entire file is written to the media. It is expected that the Entries in the Directory Section are complete and accurate. The field values of the entries where this information is not yet known should have all bits set to 1.

Files that cannot be opened for backup, should still have entries in both the Data and Directory Sections but the corresponding Data Area size(s) may be 0. In addition, the file error bit should be set in the Directory Entry.

8.1.1. DATA ENTRY

The Data Entry includes a copy of the Directory Entry, path information, data bytes (if any), and extended OS data structures (if any).

8.1.1.1. FILE SET DATA ENTRY LAYOUT

Offset,	*Length*	*Description*
0,	4	Signature = 0x33CC33CC
4,	n	File Set Directory Entry (see File Set Directory Section for entry description)
4+n,	m	Path Entry
4+n+m,	k	0 or more Data Areas

8.1.1.2. PATH ENTRY LAYOUT

A Path Entry is composed of 0 or more components of the following structure (each component is the owning name space in a mixed name space backup):

Offset,	*Length*	*Description*
0,	2	Native File System ID (see Data Description IDs for list of valid values)
2,	n	Unicode characters specifying a single hierarchical component
2+n,	2	Null separator = 0x0000 (if another path component follows)

Note that last component is not followed by a null separator (the overall length is determined by the Path Size field of the Directory Entry).

8.1.1.3. DATA AREA LAYOUT

Each Data Area for an object contains some aspect of variable length information associated with the object such as data bytes, icons, access rights, resource forks, etc. The area is preceded with a 4 byte signature followed by a Data Description Entry ID that corresponds to the Data Description Entry of the object's Directory Entry. Note that the signature and ID fields (first 6 bytes) of this area are not included in the Data Area size field of the corresponding Data Description Entry.

This structure is required even if the actual Data Area size is 0.

Offset,	*Length*	*Description*
0,	4	Signature = 0x66996699
4,	2	Data Area ID (matches corresponding Data Description Entry ID)
6,	n	Data Area Data

Except for the Vendor Specific Data Areas whose structure is defined by the vendor, the Data Area Data can fall into one of three categories listed below.

8.1.1.3.1. NULL

For the Data Description IDs that do not have any corresponding Data Area defined, the area will occupy no physical media space and the corresponding Data Area size field of the Data Description Entry will contain the value 0.

The Data Description Entry ID's that have Null Data Areas follow.

8.1.1.3.1.1. UNIX (1)

8.1.1.3.1.2. DOS (2)

8.1.1.3.2. BLOB

Other Data Description IDs correspond to Data Area's that consist of raw data bytes retrieved from the object. These areas are not to be interpreted in any way other than restored in their entirety.

The Data Description Entry ID's that have Blob Data Areas follow.

8.1.1.3.2.1. DATA (7)

The bytes are the primary data bytes of the file itself.

8.1.1.3.2.2. AFP (6)

The bytes are the resource fork of the file.

8.1.1.3.3. MULTIRECORD

Data Description IDs that correspond to file systems that provide some form of extended attributes (icons, trustee entries, etc.) may contain 0 or more variable length records called Extended OS Records (XOS Record).

8.1.1.3.3.1. EXTENDED OS RECORD LAYOUT

Offset,	*Length*	*Description*
0,	1	Extended OS Record ID
1,	8	Extended OS Record Data size
9,	n	Extended OS Record Data

Extended OS Record ID

This field is to be interpreted in conjunction with the record's corresponding Data Description ID. Note, however that the ID's 0 - 1 are reserved for the common definition of invalid record and vendor specific record respectively. Refer to a particular file system below for its list of defined values.

8.1.1.3.3.2. NOVELL 2.x (8)

XOS Record ID=2: Bindery File

This record is only valid for the root entry of a volume.

Offset,	*Length*	*Description*
0,	4	Name size (in bytes)
4,	n	Name (Unicode string)
4+n,	m	Bindery file data

XOS Record ID=3: Trustee entries

Offset,	*Length*	*Description*
		The following two fields are repeated to fill the size of the record.
0,	4	Trustee ID
4,	4	Trustee Maximum Rights Mask

8.1.1.3.3.3. NOVELL 3.X (3)

XOS Record ID=2: Bindery File

This record is only valid for the root entry of a volume.

Offset,	*Length*	*Description*
0,	4	Name size (in bytes)
4,	n	Name (Unicode string)
4+n,	m	Bindery file data

XOS Record ID=3: Trustee entries

Offset,	*Length*	*Description*
		The following two fields are repeated to fill the size of the record.
0,	4	Trustee ID
4,	4	Trustee Inherited Rights Mask

XOS Record ID=4: Volume Restrictions

This record is only valid for the root entry of a volume.

Offset,	*Length*	*Description*
		The following two fields are repeated to fill the size of the record.
0,	4	Object ID
4,	4	Restriction (in 4KB blocks)

8.1.1.3.3.4. NOVELL 4.X (9)

XOS Record ID=2: Directory Services

This record is only valid for the root entry of a volume.

Offset,	*Length*	*Description*
0,	2	Object Name size (in bytes)
2,	n	Object Name (includes full path to object) (Unicode string)
2+n,	4	Object Value size
6+n,	m	Object Value

XOS Record ID=3: Trustee entries

Offset,	*Length*	*Description*
		The following two fields are repeated to fill the size of the record.
0,	4	Trustee ID
4,	4	Trustee Rights

8.1.1.3.3.5. NT (5)

XOS Record ID=2: NTFS EA entry

Offset,	*Length*	*Description*
0,	4	EA Name size (in bytes)
4,	n	EA Name (Unicode string)
4+n,	4	EA Value size
8+n,	m	EA Value

XOS Record ID=3: NT Registry entry

These records are only valid for the root entry of a volume. The NT registry's "root" keys are stored in separate records and identified by their key handle.

Offset,	*Length*	*Description*
0,	4	Root Key handle (as defined by operating system)
4,	n	Root Key data (operating system format)

8.1.1.3.3.6. Windows 95 (10)

XOS Record ID=2: Windows 95 Registry entry

These records are only valid for the root entry of a volume. The Windows 95 registry's "root" keys are stored in separate records and identified by their key handle.

Offset,	*Length*	*Description*
0,	4	Root Key handle (as defined by operating system)
4,	n	Root Key data (operating system format)

8.1.1.3.3.7. OS/2 (4)

XOS Record ID=2: OS/2 EA entry

Offset,	*Length*	*Description*
0,	4	EA Name size (in bytes)
4,	n	EA Name (Unicode string)
4+n,	4	EA Value size
8+n,	m	EA Value

8.2. FILE SET DIRECTORY SECTION

The File Set Directory Section is composed of one or more Directory Entries concatenated together. This section is aligned on a segment boundary to allow it to be located easily during restore.

Each entry is of variable size. See File Set Data Section description for layout information.

8.2.0.1. DIRECTORY ENTRY

A Directory Entry is composed of a Fixed Portion and Data Description Portion consisting of one or more Data Description Entries.

Directory Entry = (Fixed Portion) + (Data Description Entry)...

8.2.0.1.1. FIXED PORTION

The fixed portion is always present and contains information common to all files in the backup set.

Should it become necessary to augment the information contained in the Fixed Portion for future enhancements to this logical specification, Data Description Entry ID's may be reserved with corresponding structures containing the additional information. Older versions of software would simply ignore unrecognized Data Description ID's. See the discussion of Data Description Portion that follows.

8.2.0.1.1.1. FIXED PORTION LAYOUT

Offset,	*Length*	*Description*
0,	2	Directory Entry size
2,	8	Data Entry size
10,	2	Path Entry size
12,	2	Native File System (See Data Description ID for list of valid values)
14,	1	Traversal Byte

bit 0: Directory
bit 1: Empty directory
bit 2: File error during backup (not all data retrievable)
bit 3: Last entry in directory
bit 4: Last entry in current media
bit 5: Last entry in entire backup set
bit 6: Root Entry
bit 7: Reserved (set to 0)

Directory Entry size
Value includes all fields that follow in the Fixed Portion as well as all Data Description Entries in the following Data Description Portion but does not include this 2 byte field.

Data Entry size
Value accounts for all bytes within a Data Entry (see layout of Data Entries).

value=

4 + (Signature field)
2 + (Directory Entry size field)
Directory Entry size +
Path Entry size +
4 * number of Data Areas + (Data Area signatures)
2 * number of Data Areas + (Data Area ID's)
sum of Data Area sizes (found in subsequent Data Description Entries)

OR all bits set to one, if any component cannot be determined at the time of the calculation. Note that all components should be known when the File Set Directory Section is created.

Path Entry size
Value accounts for all bytes within the corresponding Path Entry.

Native File System
Value equals one of the Data Description IDs that corresponds to a file system within which this object was created. For example, in the case of Novell, this indicates which Data Description Entry is the *creator* name space for the file.

Traversal Byte

Directory bit
When set, indicates that object is a directory.

Empty directory bit
When set, indicates that directory contains no child entries and should not be descended into while traversing.

File error during backup
When set, indicates that an error was encountered while backing up the file and some or all of the data may not be present. The size fields found in the corresponding Directory Entry's Data Description Entries within the File Set Directory Section will contain the actual sizes encountered.

Last entry in directory
When set, indicates that object is that last entry for the current directory.

Last entry in current media
When set, indicates that object is that last entry begun on the current media. Marks the end of the File Set Directory Section if multiple media are required.

Last entry in backup set
When set, indicates that object is the last entry in the entire backup set. Marks the end of the File Set Directory Section regardless of number of media required.

Root Entry
When set, indicates that object is a root or volume level entry. The name associated with the entry shall be the name of the device that the backup originated from. The root name may include an optional label for labeled source volumes, or path names for mapped or mounted devices which may also include a logical or mapped device name explicitly enclosed in parenthesis. Examples of valid root names follow:

C:	DOS device name
(C:)	DOS device name enclosed in parenthesis
MY_DISK(C:)	DOS label and device name
/usr/mounted	UNIX mounted device
\\SERVER\SYS	UNC server name
\\SERVER\SYS(M:)	UNC server name and DOS logical mapped device name

A root entry represents the upper most parent node with respect to the file entries contained on the respective device. Multiple root entries may be thought of as sibling directory entries at the same level of the file set directory hierarchy. For this reason, multiple root entries may appear first in a given file set directory with the last root entry being indicated by the Last Entry in Directory bit.

In addition to following another root entry, root entries may appear after the last file entry with a Last Entry in Directory bit indication that is associated with a root entry that also has the Last Entry in Directory bit set. In other words, once a tree hierarchy of one or more devices is complete, a new tree of one or more devices may be appended to the file set. For example, if during the course of a backup, several files have file open conflicts, the files may be retried and included again under a second root entry after the remaining files of the device have been written.

8.2.0.1.1.2. DATA DESCRIPTION PORTION

The Data Description Portion contains one or more Data Description Entries with information to determine the size and ordering of the variable length areas (Data Areas) that are included for the object within the File Set Data Section. It also provides for file attributes to be saved with the file as they pertain to the file system from which they were retrieved.

This is a repeating field to allow for mixed file system support. For example, a particular file system (such as Novell) may contain a Data Description information for a variety of supported file systems (or name spaces in the case of Novell). This method maximizes interchangeability with software packages that only support a subset of the systems provided for by this specification.

Note that Data Description ID's may be reserved in the future to provide for extensibility with respect to the current common Fixed Portion. Also, when the compatibility of a particular file system changes such that the information contained in the Data Description Structure becomes inadequate or incompatible, new ID's may be reserved for these file system revisions to either augment or replace information in the previous revision.

Data Description Entries are ordered by their corresponding ID's priority (from highest to lowest).

8.2.0.1.1.3. DATA DESCRIPTION ENTRY LAYOUT

Offset,	*Length*	*Description*
0,	2	Data Description Entry ID
2,	8	Data Area size (not including signature and ID)
10,	2	Data Description Structure size (not including this field)
12,	n	Data Description Structure (see below)
12+n,	2	Name Size (in bytes, 0 if not applicable based on Data Description ID)
14+n,	m	Name (Unicode)

8.2.0.1.1.4. DATA DESCRIPTION STRUCTURES

Data (7)

Data has a null Data Description Structure and no name applies (I.e., bytes 10 - 11 (structure size) & 12 - 13 (name size) should be set to 0).

Novell 2.x (8)

Offset,	*Length*	*Description*
0,	4	Attributes (applicable only to files) byte 0 bit 0: Read only access byte 0 bit 1: File is hidden byte 0 bit 2: File is a system file. byte 0 bit 3: Execute only byte 0 bit 4: File is a directory byte 0 bit 5: File needs archiving byte 0 bit 6: Reserved (set to 0) byte 0 bit 7: Shareable byte 1 bit 0-2: Search mode (see OS documentation for valid values) byte 1 bit 3: Reserved (set to 0) byte 1 bit 4: Transactional byte 1 bit 5: Indexed byte 1 bit 6: Read Audit byte 1 bit 7: Write Audit byte 2 bit 0: Private Directory flag byte 2 bits 1-7: Reserved (set to 0) byte 3 bits 0-7: Reserved (set to 0)
4,	1	Reserved
5,	4	Owner ID
9,	8	Archive Date/Time (applicable only to files)
17,	8	Creation Date/Time
25,	8	Modify Date/Time (applicable only to files)
33,	2	Inherited rights mask (applicable only to directories)
35,	8	Access Date/Time (applicable only to files)
43,	56	Reserved

Novell 3.x (3)

Offset,	*Length*	*Description*
0,	4	Attributes (applicable only to files) byte 0 bit 0: Read only access byte 0 bit 1: File is hidden byte 0 bit 2: File is a system file. byte 0 bit 3: Execute only byte 0 bit 4: File is a directory byte 0 bit 5: File needs archiving byte 0 bit 6: Reserved (set to 0) byte 0 bit 7: Shareable byte 1 bit 0-3: Reserved (set to 0) byte 1 bit 4: Transactional byte 1 bit 5: Indexed byte 1 bit 6: Read Audit byte 1 bit 7: Write Audit byte 2 bit 0: Immediate Purge byte 2 bit 1: Rename Inhibit byte 2 bit 2: Delete Inhibit byte 2 bit 3: Copy Inhibit byte 2 bit 4: Reserved (set to 0) byte 2 bit 5: File Migrated byte 2 bit 6: Reserved (set to 0) byte 2 bit 7: Don't Migrate byte 3 bit 0: Reserved (set to 0) byte 3 bit 1: Immediate Compress byte 3 bit 2: File Compressed byte 3 bit 3: Don't Compress byte 3 bit 4: Reserved (set to 0) byte 3 bit 5: Can't Compress byte 3 bit 6: Reserved (set to 0) byte 3 bit 7: Reserved (set to 0)
4,	1	Reserved
5,	4	Owner ID
9,	8	Archive Date/Time
17,	4	Archive ID
21,	8	Creation Date/Time
29,	4	Modify ID
33,	8	Modify Date/Time
41,	2	Inherited rights mask
43,	8	Access Date/Time
51,	4	Maximum Directory Capacity

Novell 4.x (9)

Offset,	*Length*	*Description*
0,	4	Attributes (applicable only to files) byte 0 bit 0: Read only access byte 0 bit 1: File is hidden byte 0 bit 2: File is a system file. byte 0 bit 3: Execute only byte 0 bit 4: File is a directory byte 0 bit 5: File needs archiving byte 0 bit 6: Reserved (set to 0) byte 0 bit 7: Shareable byte 1 bit 0-3: Reserved (set to 0) byte 1 bit 4: Transactional byte 1 bit 5: Indexed byte 1 bit 6: Read Audit byte 1 bit 7: Write Audit byte 2 bit 0: Immediate Purge byte 2 bit 1: Rename Inhibit byte 2 bit 2: Delete Inhibit byte 2 bit 3: Copy Inhibit byte 2 bit 4: Reserved (set to 0) byte 2 bit 5: File Migrated byte 2 bit 6: Reserved (set to 0) byte 2 bit 7: Don't Migrate byte 3 bit 0: Reserved (set to 0) byte 3 bit 1: Immediate Compress byte 3 bit 2: File Compressed byte 3 bit 3: Don't Compress byte 3 bit 4: Reserved (set to 0) byte 3 bit 5: Can't Compress byte 3 bit 6: Reserved (set to 0) byte 3 bit 7: Reserved (set to 0)
4,	1	Reserved
5,	4	Owner ID
9,	8	Archive Date/Time
17,	4	Archive ID
21,	8	Creation Date/Time
29,	4	Modify ID
33,	8	Modify Date/Time
41,	2	Inherited rights mask
43,	8	Access Date/Time
51,	4	Maximum Directory Capacity

NT (5)

It is suggested that a DOS entry be included with each File Set Entry to provide Windows and DOS interchange for basic attributes and short file name support.

Offset,	*Length*	*Description*
0,	4	Attributes
		byte 0 bit 0: Read only access
		byte 0 bit 1: Hidden
		byte 0 bit 2: System
		byte 0 bit 3: Reserved (set to 0)
		byte 0 bit 4: File is directory
		byte 0 bit 5: File needs archiving
		byte 0 bit 6: Reserved (set to 0)
		byte 0 bit 7: Normal
		byte 1 bit 0: Temporary
		byte 1 bit 1: Atomic write
		byte 1 bit 2: Transaction write
		byte 1 bits 3-7: Reserved (set to 0)
		byte 2 bits 0-7: Reserved (set to 0)
		byte 3 bits 0-7: Reserved (set to 0)
4,	8	Creation Date/Time
12,	8	Access Date/Time
20,	8	Modify Date/Time

Windows 95 (10)

These entries are optional if the following information is unavailable or unsupported by the operating system. It is suggested that a DOS entry be included with each File Set Entry to provide Windows and DOS interchange for basic attributes and short file name support.

Offset,	*Length*	*Description*
0,	4	Attributes
		byte 0 bit 0: Read only access
		byte 0 bit 1: Hidden
		byte 0 bit 2: System
		byte 0 bit 3: Reserved (set to 0)
		byte 0 bit 4: File is directory
		byte 0 bit 5: File needs archiving
		byte 0 bit 6: Reserved (set to 0)
		byte 0 bit 7: Normal
		byte 1 bit 0: Temporary
		byte 1 bit 1: Atomic write
		byte 1 bit 2: Transaction write
		byte 1 bits 3-7: Reserved (set to 0)
		byte 2 bits 0-7: Reserved (set to 0)
		byte 3 bits 0-7: Reserved (set to 0)
4,	8	Creation Date/Time
12,	8	Access Date/Time
20,	8	Modify Date/Time

OS/2 (4)

Offset,	*Length*	*Description*

0,	4	Attributes byte 0 bit 0: Read only access byte 0 bit 1: File is hidden byte 0 bit 2: File is a system file. byte 0 bit 3: Reserved (set to 0) byte 0 bit 4: File is a directory byte 0 bit 5: File needs archiving byte 0 bit 6: Reserved (set to 0) byte 0 bit 7: Reserved (set to 0) byte 1: Reserved (set to 0) byte 2: Reserved (set to 0) byte 3: Reserved (set to 0)
4,	8	Creation Date/Time
12,	8	Access Date/Time
20,	8	Modify Date/Time

AFP (6)

Offset,	***Length***	***Description***
0,	2	Attributes
2,	6	Pro DOS field (set to 0 if unsupported)
8,	8	Creation Date/Time
16,	8	Access Date/Time
24,	8	Modify Date/Time
32,	8	Backup Date/Time
40,	32	Finder Information
72,	n	Comment (comment length is reflected by the Data Description Structure size - 72)

UNIX (1)

Offset,	*Length*	*Description*
0,	3	Attributes byte 0 bit 0: Owner Read Access byte 0 bit 1: Owner Write Access byte 0 bit 2: Owner Execute Access byte 0 bit 3: Group Read Access byte 0 bit 4: Group Write Access byte 0 bit 5: Group Execute Access byte 0 bit 6: Other Read Access byte 0 bit 7: Other Write Access byte 1 bit 0: Other Execute Access byte 1 bit 1: Set user ID on execution byte 1 bit 2: Set group ID on execution byte 1 bit 3: Save text image after execution byte 1 bit 4: Linked object byte 1 bit 5: Character Device object byte 1 bit 6: Block Device object byte 1 bit 7: Regular Device object byte 2 bit 0: FIFO Device Object byte 2 bit 1-7: Reserved (set to 0)
3,	8	Creation Date/Time
11,	8	Modify Date/Time
19,	8	Access Date/Time
27,	4	Inode number
31,	4	User ID
35,	4	Group ID
39,	2	Major Device Number (if object is defined as a device)
41,	2	Minor Device Number (if object is defined as a device)

DOS (2)

It is suggested that a DOS entry accompany any File Set Entry for operating systems that support DOS file naming conventions.

Offset,	*Length*	*Description*
0,	1	Attributes bit 0: Read Only Access bit 1: Hidden bit 2: System bit 3: Archive bit 4 - 7: Reserved (set to 0)
1,	8	Modify Date/Time

8.2.1. DIRECTORY ENTRY ORDERING

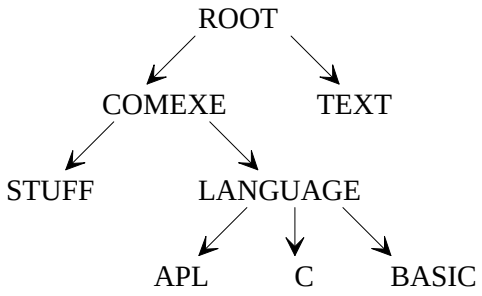
The ordering of the entries within the File Set Directory Section will be consistent with a breadth-first, preorder directory traversal of the source tree. A preorder traversal in this context is defined as visiting each entry in a given directory level prior to descending into the left-most (occurs first at parent level) child directory through right most (occurs last at parent level) child directory. Note that a node in this traversal scheme is considered to be an entire directory level (all entries) and not the individual entries.

There are two advantages to this method. First, the path alterations required to perform the entire traversal are minimized. Second, and more importantly, the data area of any complete sub-tree will be contiguous upon the tape, allowing rapid recovery of a directory and its children.

A root entry will always appear first in the directory section with all of the root directory entries following (in the order provided by the file system). Its last entry (as will the last entry of all directories) shall be indicated by the setting of the 'Last entry in directory' bit of the Traversal Byte. Following will be the first root directory entries, and below it, all of its sub-directory entries. The end of the entire directory table is indicated by the final entry having a special end-of-backup bit set. Note, that volumes spanning multiple media will have partial directories on each media with the final entry of these partial directories having a special end-of-media bit set.

Ordering example:

Given a source tree as follows:

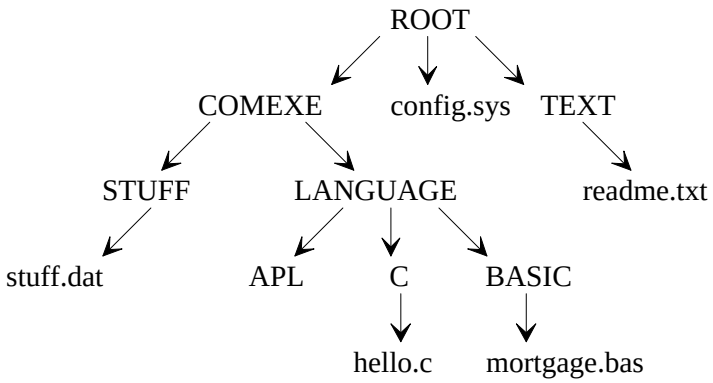


The directories would be traversed in the following order:

```
ROOT
        COMEXE
                STUFF
                LANGUAGE
                        APL
                        C
                        BASIC
        TEXT
```

Note that each directory node listed in the example above includes all of the entries at that level.

The following example shows the ordering of the individual entries (files are represented by lower case names):



Results in:

ENTRY	TRAVERSAL BYTE Directory	Empty directory	File error	Last entry in directory	Last entry in current media	Last entry in entire backup set	Reserved	Reserved
ROOT	1	0	?	1	0	0	0	0
COMEXE	1	0	?	0	0	0	0	0
config.sys	0	0	?	0	0	0	0	0
TEXT	1	0	?	1	0	0	0	0
STUFF	1	0	?	0	0	0	0	0
LANGUAGE	1	0	?	1	0	0	0	0
stuff.dat	0	0	?	1	0	0	0	0
APL	1	1	?	0	0	0	0	0
C	1	0	?	0	0	0	0	0
BASIC	1	0	?	1	0	0	0	0
hello.c	0	0	?	1	0	0	0	0
mortgage.bas	0	0	?	1	0	0	0	0
readme.txt	0	0	?	1	1	1	0	0

8.2.2.LINK SUB-SECTION

The Link Section is only included when a volume spans more than one media. It provides a means for determining what files are contained on which sequence of the previous media set. This section is a sub-section of the File Set Directory Section and its size is always included in calculations of File Set Directory Section size.

8.2.2.1. LINK SUB-SECTION LAYOUT

Offset,	*Length*	*Description*
0,	4	Signature = ASCII string 'LTLT'
4,	1	Media Sequence Number (indicates number of quadwords that follow)
5,	8	Media Ending Offset in uncompressed bytes (repeated Media Sequence # times, ordered 1st to last tape)

9. VOLUME COMPRESSION

The Volume Data Area of a compressed volume is comprised of one or more Compression Extents each of which may contain one or more Compression Frames. Additionally, if segment spanning is used (indicated by setting the segment spanning bit in the corresponding Volume Table Entry), Compression Frames are allowed to span segments. In this case each 32K segment begins with a Next Extent Offset value to enable locating Compression Extents.

9.1. COMPRESSION EXTENTS

Note that no more than one Compression Extent may start in a single segment.

9.1.1. NON-SEGMENT SPANNING COMPRESSION EXTENTS

If segment spanning is not used, it is expected that each segment will contain exactly one Compression Extent aligned at byte 0 of the segment. Also, Compression Frames are not allowed to span into a subsequent segment.

9.1.2. SEGMENT SPANNING COMPRESSION EXTENTS

If segment spanning is used, the first two bytes of every segment contain a Next Extent Offset to enable locating Compression Extents within the segments. The value of this little endian field is an offset to the next Compression Extent found within the current segment (if any). If the entire segment is a continuation of a previous Compression Extent, the Next Extent Offset will have a value of 0. Also note that the hi-order bit of this field does not indicate compressed/uncompressed data as it does in the Frame Size field described below.

Next Extent Offset	**Indicates the following**
2	the following byte (segment offset 2) is the start of a new Compression Extent (this would be the case of the initial Next Extent Offset found in the first segment of the volume)
>2	the bytes following are the continuation of a Compression Frame and the Next Extent Offset is an offset within the current segment to the next Compression Extent
0	the following bytes are the continuation of a Compression Frame that may also continue into the next segment as well

9.1.3. COMPRESSION EXTENT LAYOUT

The form of a Compression Extent is as follows:

Offset,	***Length***	***Description***
0,	8	Uncompressed Volume Byte Offset
8,	n	Compression Frame(s) (may be repeated as necessary to fill segment)

Uncompressed Volume Byte Offset

The Uncompressed Volume Byte Offset provides a means to seek to individual files without decompressing preceding segments. The Uncompressed Volume Byte Offset can be described as the sum of the number of uncompressed bytes in all preceding segments. For example, in the first segment

this value would be 0, in the second it would be the uncompressed bytes found in the first segment, in the third it would be the sum of the uncompressed bytes of the first and second segments, etc.

When the Directory Last feature is used, the Uncompressed Volume Byte Offset shall be reset to zero when the File Set Directory is written. Further, in the case of a linked volume, when the File Set Directory is written at the end of each media in the linked set and when writing of the File Set Data is resumed at the start of the next media in the linked set, the Uncompressed Volume Byte Offset shall continue as if the data immediately follows the data written on the previous media. That is, the area to which the File Set Directory was written shall not be included in the totalization of the Uncompressed Volume Byte Offset. This allows for seek algorithms to be implemented in a much more straightforward fashion when linking, compression and directory last are employed.

9.1.4. COMPRESSION FRAME

The form of a Compression Frame is as follows:

Offset,	***Length***	***Description***
0,	2	Frame Size (or portion thereof contained wholly within current segment, whichever is smaller. Does not include this field.) Also used to indicate uncompressed data (see description below).
2,	n	Data Bytes (compressed or uncompressed volume data bytes)

Frame Size

Frame Size is equal to *n* and has a value in the range of 1 through 32K-ECC-2-frame offset within segment (ECC represents the number of bytes reserved at the end of the segment for Error Correction Code). For example, if a frame begins at the 100th byte of a 32K segment that has 3K reserved for error correction code, its Frame Size could be calculated as follows:

Frame Size = 32K - 3K - 2 - 99

The hi order bit of the Frame Size is used to indicate whether the Data Bytes are in compressed or uncompressed “raw” form. If the hi order bit is set, the Data Bytes are uncompressed; if clear, they are compressed.

Note, that if a frame spans a segment in a segment spanning volume, the Frame Size only accounts for the bytes contained in the current segment. The size of the remaining portion of the frame is implied by the Next Extent Offset of the subsequent segment.

Data Bytes

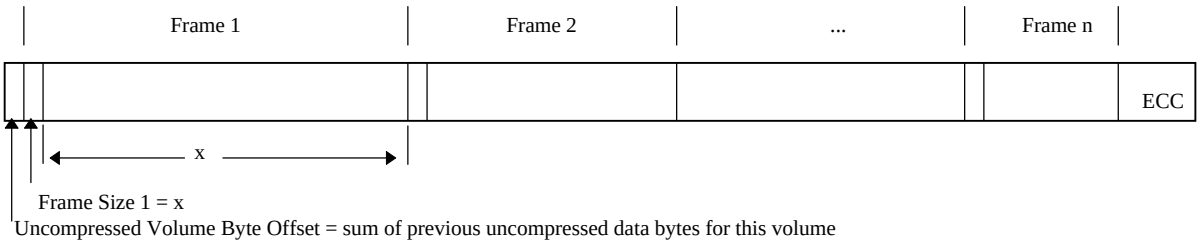
The Data Bytes are the compressed or uncompressed bytes that map to the next sequence of Volume Data Area bytes for the volume. A maximum of 63,488 raw bytes may be compressed into a single Frame.

Compression Frames may be concatenated together to fill a segment. However, if a frame ends within the last 18 bytes of available space in a segment (after accounting for ECC), these remaining bytes are to be null filled and remain unused. If a given frame has a Frame Size that leaves more than 18 bytes available, another frame is expected to immediately follow.

9.1.5.EXAMPLE SEGMENT OF A NON-SEGMENT SPANNING VOLUME

Entire segment (excluding ECC) is a single Compression Extent consisting of n Compression Frames.

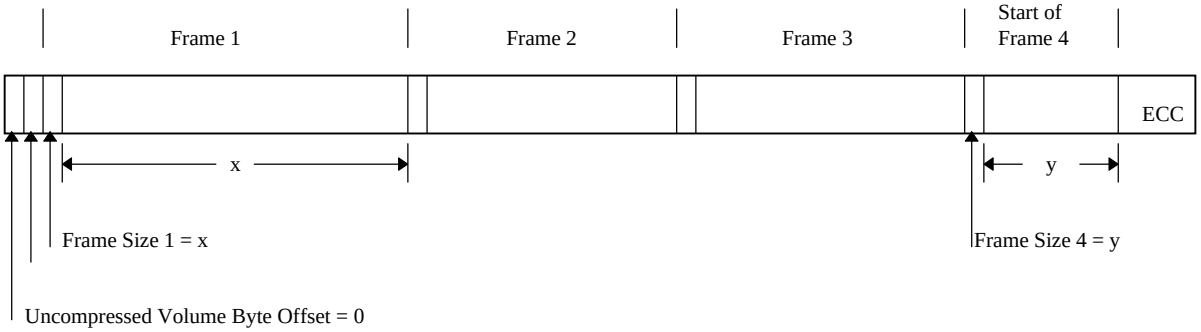
Segment i



9.1.6.EXAMPLE SEGMENTS OF A SEGMENT SPANNING VOLUME

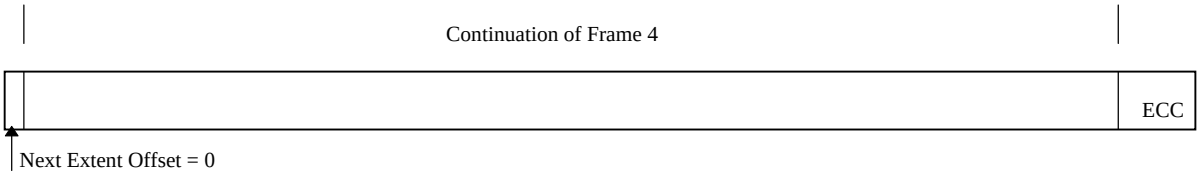
The first Compression Extent consists of the first 4 Compression Frames and occupies the entire first two segments (excluding ECC) and part of the third segment as well.

Segment 1



Next Extent Offset = 2

Segment 2



Segment 3