ISO 9660 File System Forensic Analysis

Version 1.3

Anders Thulin, 2021-09-04

This is an attempt to describe the ISO 9660 file system from a forensic perspective. It is patterned on the approach used by Brian Carrier in his book *File System Forensic Analysis* (Addison-Wesley, 2005), referred to a *FSFA* below. As ISO 9660 is a fairly complex file system in all its details, using a well-known pattern may hopefully make it easier to approach.

One of the more important goals of forensic analysis of file system is that of being able to interpret the file system and file structures so that no significant file system entity or metadata content remains unidentified or is incorrectly or incompletely interpreted. Other aspects deal with how user activity can be identified from file system metadata, or how different implementations leave traces of its particular use of the file system. ISO 9660 is easier to analyze in that by itself it leaves no traces of such user activities.

Other aspects of the file system are not necessarily treated in detail, and may only be indicated by a reference to relevant parts of the standard itself.

ISO 9660 is specified by the International Organization for Standardization (ISO). The first specification was:

> ISO 9660:1988
>
> *Information processing — Volume and file structure of CD-ROM for information interchange*

Two editions were published in 1988: the first on 1988-04-15, and a corrected edition on 1988-09-01. In this document, any references to ‘ISO 9660:1988’ always refer to the second, corrected edition.

The first update to the standard document was issued in May, 2013:

> ISO 9660:1988/Amd. 1:2013
>
> *Information processing — Volume and file structure of CD-ROM for information interchange, Amendment 1*

It contains one major addition to the standard: that of the Enhanced Volume Descriptor, which specifies relaxed requirements on file name length, depth of directory hierarchies etc. that other volume descriptors impose. The document itself, however, contains only the changes made to ISO 9660:1988, and is thus inconvenient to use for reference.

The second update was issued in April, 2020:

> ISO 9660:1988/Amd. 2:2020
>
> *Information processing — Volume and file structure of CD-ROM for information interchange, Amendment 2*

This amendment does not add any technical information: it adds a number of headings or subheadings to the earlier text, on the same pattern as Amendment 1.

The amendment does provide what may be presumed to be an official link to a web page where the Joliet specification can be found. Joliet is mentioned in Annex B.2: the link is to the ISO Standard Maintenance Portal at:

> *https://standards.iso.org/iso/9660/ed-1/en/amd/2*

This document, however, is different from the original document which was published as part of a Windows 95 DDK around May, 1995. (The original document is a Word document, and, for example, the subsections *Overview* and *Terminology and Notation* are independent chapters, not subsections of the *Preface*. Other changes may also have taken place, so this web document should probably be used with some caution, and preferably be collated with the original document.)

ECMA International has also published related standards. The current version is:

> ECMA-119
>
> *Volume and File Structure of CDROM for Information Interchange* (3rd edition, December 2017)

which, according to its Introduction, “is technically identical with ISO 9660 updated with the Amendment 1 of 2013-05-01”. ECMA-119 can be downloaded for free from the ECMA International web site at www.ecma-international.org .

Additional known related standards or specifications are referenced in the last chapter.

ISO 9660 Concepts

While ISO 9660 media have often been used as equivalents for floppy disks and similar storage and interchange media, the standard as a whole is considerably more complex. It allows multiple generations of a file stored on multiple volumes to be identified and accessed, it allows for file data to be organized by different record structures, by interleaving and encoded by different character sets, allows file access to be restricted by date/time, permits metadata about publisher, and bibliographical references to be added, and there are opportunities to include system- or application-dependent data or processing at several levels in the file structure hierarchy. At the same time, many of these capabilities can only be used if the publisher and the recipient agree on their use – a situation not present in most other file systems, and not necessarily supported by modern tools for creation or examination of ISO 9660 volumes.

Chapter 2 in the standard states conformance requirements both for ISO 9660 volumes themselves and for the information systems that create and read such volumes. This is where interpretation of ISO 9660 starts: if a particular volume does not conform to these requirements, knowledge of external factors (such as implementation details) will be required to interpret volume content correctly, and if a particular receiving system diverges from standard, discussion and interpretation of volume, directory and file access control effects will be more complex. This document does not in general cover such issues, although some of them may be mentioned in passing.

Note

There seems to be no rationale or description of intended use of the standard.

Author's note: My current take on ISO 9660 is that it is best understood as a framework intended for larger companies or organizations for publishing information intended for their customers, particularly in situations when changes or updates will be published on a fairly regular basis, with a known lifetime, and with a need to be managed and traceable over that lifetime of the information.

Examples of such use are documentation and reference manuals for major systems such as telephone exchanges, trains, aeroplanes, power plants and more. Other fields of use may be regulations, standards, or distribution of official or public data sets, for example census data.

That is, ISO 9660 appears to be more focussed on the needs and requirements of periodically issued information from a publisher/manufacturer to its customers, than single volume publication intended for more-or-less random recipients such magazine or game buyers, or as a means of *ad hoc* file exchange between end users.

Expressed otherwise, I believe the subtitle of the ISO 9660 standard (which is '*Volume and file structure of CD-ROM for information interchange*') to be somewhat misleading. A more appropriate subtitle would, in my opinion, have been '*Volume and file structure of CD-ROM for information publication.'*

The Joliet extension by Microsoft changed some of this, as it allows for more informal relation between originator and recipient of a volume when character code sets do not need to be agreed on

in advance. (See additional information at the end of this document.) Late changes in ISO 9660 may also impact this use.

Conventions

In this document, references to the standard are on the form '[ISO 9660, 1.2.3.4]', which refers to section 1.2.3.4 of the latest version, *i.e.* the version established by the 2020 amendment. (As this amendment does not change any technical information, references may in general also be taken to refer to the text established by the 2013 amendment.) References to specific versions of the standard include additional information.

The ISO 9660 standard, as well as this document, uses the format '(12)' or '(12 34 56)' to represent the value of one or more individual bytes in hexadecimal form [ISO 9660, 5.1].

In cases where non-byte-sized values are used or discussed, this document uses conventional hexadecimal notation with '0x' prefix (*e.g.* 0x123456) in cases where the binary content of the value is relevant.

Data representation

Integers

ISO 9660 stores different forms of integers in 1, 2, 4 and 8 byte fields [ISO 9660, 7.1–7.3]. Normally unsigned representation is used. Signed single-byte integers are used only for time zone offsets from GMT in time stamps.

16-bit and 32-bit integers may appear in three formats: with least significant byte first, with most significant byte first, or in both-byte order:

Value	Least Significant Byte First	Most Significant Byte First	Both-Byte Order
0x1234	(34 12)	(12 34)	(34 12 12 34)
0x12345678	(78 56 34 12)	(12 34 56 78)	(78 56 34 12 12 34 56 78)

Note that the both-byte order provides both LSBF and MSBF formats in one field, and so requires twice as many bytes to store, as well as additional validation checks that the first half of the field expresses the same value as the second half.

Strings

ISO 9660 was designed for 8-bit character string data. Support for wider character may be possible, but will usually cause interpretation issue over the meaning of the length of a string: the number of characters in the string, or the number of bytes it occupies?

ISO 9660 has two main string representation formats: dynamic-length and fixed-length string fields.

Dynamic-length string fields are used in situations where storage space is limited, such as directory records or path table records. They are associated with a string length field, and do not contain any unused data.

Fixed-length string fields are used in situations where adequate storage space is available, for example in volume descriptors. In these cases, a maximum allowable string length is defined and allocated, and the characters of the string are stored, one by one, and finally padded to the end of the field by a special byte value known as FILLER [ISO 9660, 7.4.3.2, 7.4.5]. FILLER bytes at the end of a fixed-length string field are not interpreted as string data.

In fields related to primary and supplementary volume descriptors, the standard defines FILLER to be the character (20) which also may be part of normal string data when it does not appear at the end of the field. In enhanced volume descriptors, the FILLER character is agreed on between the originator and recipient of the volume, but is not documented anywhere in the volume itself. It may in some circumstances be inferred from the presence of multiple identical trailing bytes in relevant descriptor fields in cases, but such inferences cannot be completely conclusive.

(Some strings fields in volume descriptors also allow for data *longer* than the actual string field. They use the character (5F) in the first position of the field to indicate that the remaining field contents should be interpreted as a root directory file identifier, and in which file the actual information can be found.)

Practical Notes

Some creation software fails to follow these requirements, and add (00) characters at the end of fixed-length string data, or terminate strings with a single NUL byte, sometimes followed by apparently random byte data. (Note that enhanced volume descriptors may use (00) as their FILLER character, and that the widely-used Joliet format does stipulate (00) as FILLER.)

Unusual choice of FILLER may cause problems in forensic interpretation of enhanced volume descriptors.

Character sets

ISO 9660 defines a mandatory character set for file and directory metadata (including file and directory names) in the primary volume descriptor and its related directory hierarchy. This is a subset of ISO 646 IRV [ISO 9660, 7.4, Appendix A].

This character set may also be used for supplementary and enhanced volume descriptors by omitting to specify any other character set [ISO 9660, 8.5.6].

ISO 9660 also allows file data to be encoded in character sets that are closer to the recipient’s needs, for example in platform-related character sets, or in national character sets such as Greek or Russian. Support of any such character set is not mandated by the standard, but requires the volume originator and its recipient to agree on character coding. This means that either the ISO 646 IRV subset has to be used with its limitations, or that a more complex relationship is needed, for example one where the originator not only creates the volume but also recommends or even

prescribes what ISO 9660 software must be used by the recipient to access the volume, and possibly also how that software needs to be configured.

ISO 9660 bases its support for additional character sets on the ISO 2022 coding standard [ISO 9660, 8.5.6 9.5.18]. This is a fairly complex standard, and will only be described here to the extent necessary to understand simple ISO 9660 use. For full understanding, it is necessary to refer to the standard itself.

ISO 9660 allows the originator to specify G0 and G1 character sets for encoding file and directory names and additional metadata in supplementary and enhanced volume descriptor hierarchies. These are approximately ‘the lower and upper half of an 8-bit code table’. G0 refers to printing characters associated with code points 0x21–0x7E (94 characters), while G1 refers to printing characters with code points in the range 0xA1–0xFE (94 characters) or 0xA0–0xFF (96 characters). For example, one use case might have ANSI/ISO 7-bit code as the G0 character set, and leave G1 unspecified, while another may add upper Latin-1/ISO 8859-1 as G1 for additional Western European character support.

The actual selection of G0 or G1 character sets is decided by escape sequences in the ISO 9660 volume descriptor. They are usually identified by pre-registered escape sequences, although unregistered (*i.e.* implementation-dependent) escape sequences may also be used. The register of such code sets is part of *ISO International Register of Coded Character Sets to be Used With Escape Sequences*. (Note: Some of the escape sequences listed in that document cannot formally be used with ISO 2022 and thus not with ISO 9660. This includes Unicode-based character sets, and may affect Joliet usage in very strict implementations. The document can be found on the ‘net.)

ISO 9660 does not allow control characters (C0 or C1) to be selected in this manner in file and directory identifiers, but places no explicit restrictions on specifying control character sets for encoding of file data.

Practical Notes

Not all ISO 9660-reading systems accept the default character set for Supplementary Volume Descriptors.

In DOS and early Windows days, the MSCDEX utility provided ISO 9660 interpretation. Some (?) releases of MSCDEX provided a command line switch ‘/K’ that was used to indicate that shift-JIS Kanji character set was used for file names: it is not known how that related to escape sequences specified on the volume itself.

ISO 9660 Volume Analysis

ISO 9660 does not permit a physical volume (*i.e.* a CD-ROM or similar storage medium) to contain multiple ISO 9660 file systems within the same track. That is, there are no traditional partitioning structures to be identified and examined. (While there are entities called volume partitions in ISO 9660, their use is entirely system dependent [ISO 9660, 8.6].)

ISO 9660 does not obviously exclude the possibility that an ISO 9660 file system may coexist with another file system on the same medium (on CD medium, within the same track). Fairly well known and observed possibilities are FAT + ISO 9660, UDF + ISO 9660, and HFS + ISO 9660.

Identifying and handling this type of situation is not considered to fall within the scope of this document.

ISO 9660 does allow for multiple ISO 9660 volumes to make up a kind of spanned multi-disk file system. This is referred to as a Volume Set. A volume that belongs to a volume set must include the information of directory trees and path tables of *all* preceding volumes in the set.

A volume set is identified by a Volume Set Identifier, which all volume set members must use. All volumes in the volume set must be numbered consecutively, starting from 1. This is the Volume Sequence Number.

Additionally, all directory records include information of the volume sequence number of the volume where the respective file is stored. This allows a directory entry on, say, volume 5, to refer to a file or directory on volume 1. It may also allow for a volume descriptor on volume 6 to have its root directory on volume 4.

Schematic Example:

```
Volume 1: “VolumeSetA”
  Root directory
    File entry: LICENSE.TXT;1 → (local file location and size)

Volume 2: “VolumeSetA”
  Root directory:
    File entry: LICENSE.TXT;2 → (local file location and size)
    File entry: LICENSE.TXT;1 → (volume 1 file location and size)
```

This example shows how a file on a later volume can be updated. If no update is necessary, volume 2 can still refer to the original LICENSE.TXT;1 on volume 1.

There is also the concept of a Volume Group, which is one or more volumes established at the same time, but is not described here. See the standard text for information [ISO 9660, 6.6].

Practical Notes

While it is extremely unusual to find multi-volume CDs, the volume sequence number of every ISO 9660 volume under examination should be checked to ensure that the rare case when they do appear is not missed. If the volume sequence number is 2 or larger, it may indicate that complete retrieval of data from that particular volume may require that some or all earlier volumes in that volume set also are available. The Volume Set Size attribute also provides related information. In typical end-user-produced CDs both of these will be ‘1’. Best would be if these checks were performed automatically as part of all ISO 9660 analysis tasks.

Similarly, the ability of the analysis tool (or the analyst) to handle file reference from one volume to a previous volume correctly should be verified, again for the very rare case where such references may appear. Such references may only concern a section of a file: it may be complex and error prone to manually extract data correctly if the tool in use is unable to do this job. In any case, such cross-volume references should be possible to flag as anomalies.

ISO 9660 File System Analysis

Overview: Logical Sectors and Logical Blocks

ISO 9660 is described in terms of allocatable units: Logical Sectors, and Logical Blocks. Some parts of an ISO 9660 volume are required to be allocated and addressed as logical sectors (such as the system area, the volume descriptor set and, partially, directories), while other are allocated and addressed as logical blocks (such as files and directories, including any extended attribute records).

While this may cause some confusion, as long as logical block size = logical sector size it does not lead to any practical problems. This is the case for all or practically all ISO CDs in circulation. However, if logical block size < logical sector size, it becomes important to keep block addresses separate from sector addresses.

A Logical Sector [ISO 9660, 6.1.2] typically corresponds to the bytes recorded as data in a (physical) sector. ISO 9660 requires a logical sector to be at least 2048 bytes (corresponding to the 2048 bytes of a CD-ROM data sector), but allows it to be any larger power of 2 (*i.e.* 2^n, where $n \geq 12$: 2048, 4096, 8192, etc.) as may be suitable. (If physical sectors are smaller than 2048 bytes, they are concatenated to make up a logical sector of at least 2048 bytes: it would be possible to implement ISO 9660 on a floppy disk by letting one logical sector = 4 physical 512-byte sectors). Information about the logical sector size of a particular volume is not stored anywhere within that volume or volume set: it must be obtained or inferred in other ways, *e.g.* from the system platform, or from system driver documentation. Logical sectors are used for allocating the system area, volume descriptors and directories.

Logical Blocks [ISO 9660, 6.2.2] are parts of logical sectors and used for allocation and addressing of files, path tables and also in defining volume partitions. (They are also used for addressing directories, which are allocated as if they were logical sectors.) Logical blocks smaller than a logical sector allow files to be allocated on sub-sector boundaries, and so help reduce unused space within a volume. ISO 9660 requires a logical block to be at least 512 bytes, but allows it to be any larger power of 2 up to the size of the logical sector. Thus, a typical CD-ROM volume, with a logical sector size of 2048 byte, may use logical blocks of sizes 512, 1024 and 2048 bytes. The size of a logical block used by a volume is recorded in volume descriptors on the volume. Within a volume set, logical block size is not allowed to change.

Logical block size is (or should be) under the control of the person who is responsible for the technical creation of the CD, a.k.a. the ‘data preparer’ [ISO 9660, 12.3.1].

Practical Notes

Some existing ISO 9660 software appear to be incapable of handling logical block sizes different from the default logical sector size of 2048 bytes.

Use of unusual logical sector size might possibly be used as obfuscation, intended to make it difficult for an ISO volume recipient to use other access software than that provided by the originator. This might lead to strictly correct ISO 9660 volumes being side-lined in an investigation

as ‘probably damaged’ if the analysis software fails to recognize them. The probability for encountering such volumes is minuscule if use of consumer ISO-creation tools can be assumed.

File System Category

This section deals only with data structures and content that are needed to allow the full file tree(s) of ISO 9660 to be traversed and interpreted for forensic analysis. This excludes items such as permission structures, and file content time restrictions, which may restrict normal user access; these are described in following sections.

The first 16 logical sectors of each volume (numbered 0-15) are known as ‘System Area’ and are undefined by the standard [ISO 9660, 6.2.1]. The System Area cannot be use for any information specified by the standard: files and directories cannot be stored here without breaking conformance to the standard. Any use of System Area is expected to be ‘documented’ by the System Identifier field in volume descriptors.

The remaining space of the volume is known as Data Area, and contains everything else: volume descriptors, path tables, directory hierarchies, volume partitions and file data.

Volume Space Size (at least System Area size + Data Area Size) is recorded in directory-tree related volume descriptors [ISO 9660, 8.4.8].

Note

Volume Space Size is expressed as number of logical blocks. In the case where logical block size is not the same as logical sector size, volume space size might not be an integer number of logical sectors. This may cause problems.

Volume Descriptors

The content of a volume file system is described by a series of Volume Descriptors that starts in logical sector 16, and is terminated by a Volume Descriptor Set Terminator [ISO 9660, 6.7.1 and subsections]. The standard does not impose any order of the sequence of other volume descriptors: while one volume may have its primary volume descriptor in logical sector 16, another volume may not.

Each individual ISO 9660 volume must have at least one directory hierarchy (rooted in the primary volume descriptor), but may have additional ones. The mandatory directory hierarchy uses a reduced ISO 646 character set for all file and directory names, as well as other metadata: all recipient systems and applications are required to handle this character set. Additional directory hierarchies are free to use other character sets, but then only subject to agreement with the recipient of the volume.

The volume descriptors that specify a directory hierarchy additionally identify the root directory of each separate hierarchy. The file-related volume descriptors are:

Primary Volume Descriptor	mandatory	implicitly specifies ISO 646 subset character set for file names and metadata
Supplementary Volume Descriptor	optional	specifies other character sets for file names and metadata
Enhanced Volume Descriptor	optional	specifies other character sets, with laxer requirements on file names and other directory tree parameters

Additional volume descriptors are:

Boot Record	optional	specifies information for a boot system identified in the record itself
Volume Partition Descriptor	optional	identifies volume partitions, again for systems identified in the record itself
Volume Descriptor Set Terminator	mandatory	marks the end of the volume descriptor set

The standard allows primary, supplementary and enhanced volume descriptors and the volume descriptor set terminator to be recorded multiple times

Practical notes

While not stated explicitly, multiple recordings of some volume descriptors is probably intended to lessen the risk that read errors in these sectors may prevent access to the part of the volume described by that volume descriptor.

The standard does not say if and how the ‘priority’ between primary volume descriptor and supplementary or enhanced volume descriptors is handled, that is, if there are multiple file-related volume descriptors present, which one should be used, or how should collisions any between paths and files in the different file trees be resolved?

As the originator of the volume is likely to recommend, or more probably stipulate, the reading system that the recipient should use, this is not an issue in what is believed to be intended use.

The names of volume descriptors may be somewhat misleading. A ‘primary volume descriptor’ is not necessarily the main or most important volume descriptor, but it is the one that is used if everything else fails. It may be better considered as ‘the default volume descriptor’. Similarly, a ‘supplementary volume descriptor’ is typically the volume descriptor that is closest to user requirements in terms of character sets (and thus language use), and so the best volume descriptor to use to meet a particular user’s needs and requirements.

Directories

Directories [ISO 9660, 6.8 and subsections, 9.1 and subsections] are ordered collections of directory records, which contain information about file or directory identifiers, logical block numbers to the referenced file or directory extents, status flags, time stamps, etc.

While directories are addressed by logical block numbers, they must be allocated on logical sector boundaries, *i.e.* they always start in the first logical block that is contained in a particular logical sector. Directories also always occupy full logical sectors, even if their actual content is shorter.

An unusual feature is that directory records are never split over sector boundaries. If a directory record does not fit in the remaining space of a directory sector, the remaining space of that sector is set to (00), and the record is instead placed in the following sector. A Directory Record must also contain an even number of bytes, and allows for the use of a special padding field to ensure this.

Directory records are ordered by a) file name, b) file name extension, c) file version number, d) associated file flag, e) the order of the file sections of the file [ISO 9660, 9.3]. The last requirement ensures a definite and predictable record order, though it implies that sections of multi-section files must be recorded in ascending order of block addresses.

The first two directory entries of all directories are special [ISO 9660, 6.8.2.2]. The first one must be a directory record with the directory name (00), and the second one a directory record with the name (01): they identify the current directory, and the parent directory, respectively. (The root directory identifies itself as its own parent directory.)

In the example below ‘(00)’ and ‘(01)’ are the special, one-byte identifiers.

Schematic Example:

```
Volume:
  Root directory @ LBN 100:
    Directory entry (00) → LBN 100    (points to self)
    Directory entry (01) → LBN 100    (root directory has self as parent)
    Directory entry “SUBDIRECTORY” → LBN 200

  Directory @ LBN 200:
    Directory entry (00) → LBN 200    (points to self)
    Directory entry (01) → LBN 100    (points to parent)
     … any additional entries
```

Unclear issues

System Use. Directory records contain a System Use field, left unspecified by the standard and intended for use by the receiving system. However, the field that identifies the target system that interprets the contents of this field is located in an extended attribute record [ISO 9660, 9.5.1]. This appears to add an overhead space cost of at least one logical block for every time the system use field of a directory record is used. This seems unnecessarily wasteful for mass use; it seems more suitable for occasional use only. More probable in practice seems to be that the System Use data in this record is not identified in the same way as System Use elsewhere, but as this may lead to faulty interpretation it must be regarded as a possible source of problems.

Enhanced Volume Descriptors. In enhanced volume descriptor directories there are no file name extensions or version numbers. The standard does not explicitly say how this absence should be interpreted when directories are sorted; this may be an area where different interpretation of implementers may occur, adding difficulty to file system interpretation.

Terminology. A similar issue affects records for directories in primary and supplementary volume descriptor hierarchies: the terms used (file name, file name extension, and file version number) do not, strictly, apply to directories; this may cause differences in interpretation among different implementers. For example, it seems possible to interpret section 9.3 to mean that file records in a directory are sorted separately from directory records.

Practical Notes

Directory Length. The length of a directory, as recorded in a directory record, must always be a multiple of the logical sector length: it says how many bytes are allocated to the directory, not how many bytes are actually used for the directory records. The end of a sequence of directory records is identified by (00)-filled bytes.

However, not all originating systems follow this rule, and many cases of incorrect directory length have been observed in the wild.

Files

The simplest ISO 9660 File is a single data stream (also known as a file section = the part of a file recorded in one extent) identified by a single directory record. The size of such a file section can be at most 2^{32} bytes. All bytes are allocated in consecutive logical blocks in the data area of the volume.

A file may also consist of multiple file sections (a.k.a. a multi-extent file), where each section is identified by a separate directory record, all of which use the same file identifier, and have special status flags identifying the record as being part of a multiple-section file. The order of these directory records decide the order in which the file sections should be accessed. However, the requirements on how directory records are sorted require these sections to be allocated in ascending order by logical block number. That is, if the first file section of a multi-extent file is allocated at logical block 500, later file segments must be allocated at the same or higher logical block numbers for strict ISO 9660 conformance. Use of multi-extent files allow files to exceed the 2^{32} byte length limit of single-extent files.

An ISO 9660 file may also, independently of any multiple file sections, have an Associated File, which effectively creates a separate byte stream as part of the File. Associated files are, like file sections, identified by separate directory entries using the same file identifier as the main file, as well as a special status flag identifying the file section(s) as being part of an associated file. The standard does not specify any interpretation of Associated File presence or contents, and does not even require that a receiving system can make such files or their data available to the user.

File sections can additionally be recorded in interleaved mode. This divides the file section data into separate File Units, each of which is separated from any preceding and succeeding file unit by an Interleave Gap containing data that is not part of the file. (For non-interleaved files sections, the terms 'file unit' and 'interleave gap' do not apply: a non-interleaved file does *not* consist of a single file unit that contains a file section.)

This does not create new file extents: the non-interleaved file data is just padded by interleave gaps, increasing its size, but the file extent remains a single extent.

If an interleaved file section is associated with an Extended Attribute Record, its file unit size must be identical to the length of the extended attribute record.

File sections can be shared or reused between files, including the file itself [ISO 9660, 6.5.1].

For interleaved files, file units and interleave gaps can be also shared between file sections, according to the standard [ISO 9660, 6.4.3.2].

Unclear issues

Why interleaving? It is not clear why anyone would make file data interleaved. It may be an attempt to provide for files that require extra computation between groups of sectors during the time the CD medium rotates for the next sector to be read. (Similar approaches have been used for floppy disks; see https://en.wikipedia.org/wiki/Interleaving_(disk_storage) for further information.)

Once sufficiently fast CD-readers had developed, it probably was more efficient to read larger chunks from the CD into temporary buffer storage before accessing it, and so eliminate file interleaving issues entirely.

Interleave gaps. There seems to be no requirements as to the content of interleave gap areas. As file sections can be reused (see below), it might be possible to place other short file sections in interleave gaps. This will almost certainly be a point on which implementations may differ.

Reuse. The standard allows the logical blocks that make up an extended attribute record to be assigned to a different file section (ISO 9660, 6.4.4.1, note 6).

It is not clear how this could be done, except in the special case that the different file section is a part of the original one.

The standard allows the logical blocks that make up a file unit and/or an interleave gap that is part of an interleaved file section to be assigned to a different file section [ISO 9660, 6.4.3.1 note 4; 6.4.3.2 note 5].

The standard allows a file section to be used more than once in the same file. From a forensic point of view, this does not create any major problems. However, the situations in which such reuse can be applied appear to be rare, as the logical block number of a segment is part of the directory sorting specification. This seems to lead to reuse being possible mainly in situations where the uses of the shared sections are ordered correctly.

Empty files. Allocation and handling of empty files is a special case, not mentioned by the standard, and thus possibly subject to different interpretation by implementers.

All directory records must contain the address of the first logical block allocated to the file or directory. The 'obvious' method of using a logical block number of 0 for files of length 0 is not mentioned as being allowed, and as logical block 0 is located in the system area of the volume, it is not allowed to be used for file sections. It seems that directory entries for empty files must refer to a 'real' logical block within the Data Space of a volume.

As logical block size usually is 2048 bytes, this could be very wasteful if many empty files were present in a volume. Use of short logical block size would mitigate the problem to some extent, but it would not entirely solve it.

If the standard reference to a file section being part of more than one file can be applied also to empty file sections, empty files can be managed by allocating a single logical block, which is then shared among all empty files on the volume. However, it is not clear from the standard that empty file sections are allowed, so this is probably a point where interpretations may differ.

Practical Notes

Empty files. ISO 9660 directory records have been observed to refer to non-empty blocks for empty file entries, such as the immediately following logical block. As the empty file has a file length of 0, it can technically refer to any logical block within the Data Space of the volume without interpretation problems, as none of the data content of the referenced block will actually be read. Not all forensic tools are happy about this, and some warn about overlapping files, damaged file structures or similar issues. This may cause confusion and not impossible require considerable additional explanation in a forensic presentation for a non-technical audience.

In some cases, directory records of empty files have been found to refer to logical block 16 (assuming logical block size = logical sector size), which is where the first volume descriptor is recorded. Again, as the directory record specifies a length of 0 bytes, the actual contents of the referenced logical block is irrelevant as it will never be read. It is, however, counter-intuitive to find a file record identifying a volume descriptor block for data, and this may cause warnings or error messages from a forensic tool, and confusion in a forensic analyst.

```
Volume:
  Root directory @ LBN 100:
    File entry "EMPTY.TXT;1"      → LBN 100 (size 0)
    File entry "ALSO_EMPTY.TXT;1" → LBN  16 (size 0)
```

The file “EMPTY.TXT;1” refers to the root directory itself for (empty) file data.
The file “ALSO_EMPTY.TXT;1” refers to the volume descriptor recorded at logical block 16 (we assume logical block size = logical sector size).

Path Tables

Path tables summarize directory path information. Instead of traversing a file path by reading each directory individually until the final directory in the path has been reached, the recipient system can use the path tables. A path table contains pre-assembled information of the name and location of each directory in the directory tree, and so provides a way of accessing the final directory of a path directly once the path table has been read into memory.

Path tables come in pairs of one Type L path table, and one Type M path table. In Type L path tables all numerical values are stored in LSBF format, while in Type M tables, MSBF format is used.

One pair of path tables is mandatory; a second pair may be specified. They should contain the same information, only recorded in different extents.

Path tables are sorted [ISO 9660, 6.9.1] by the level of the directory (root directory has level 0, its subdirectories level 1, and so on), by the path table entry number of the parent directory, and finally by the target directory identifier.

Path tables are referenced from the file-hierarchy volume descriptor.

Unclear

Path table sizes. There is no explicit restriction on the size of a path table, apart from the 32-bit byte length recorded in the volume descriptor. But there may be an indirect restriction.

The entries in a path table contain the directory numbers of the parent directories of each entry. This directory number is the index of the parent directory in the path table, and is stored as a 16-bit number, starting from 1, which indicates the root directory.

Thus, path table records can only refer to parent directories located in the first 65535 entries. This appears to limit the extent to which a volume set file tree can grow: once the limit has been reached, it is not possible to add more parent directories.

If a volume contains 65535 path table entries, and one additionally directory is added ‘at the end’ (assuming that it actually gets sorted into the last position), it would be numbered as 65536. But if it does not contain a subdirectory, that entry number does not require to be recorded anywhere. This seems to allow for additional directories to be present, as long as they can be ensure to be assigned an index number > 65535.

It is difficult to say if this interpretation is against the standard text or not: it probably cannot be relied on to be used in just any implementation. And it seems to require special file tree design to ensure that these ‘extra’ directories are sorted after the 65535th entry – they need to be located in the ‘deepest’ directory in the file tree.

Multi-volume volume sets may also cause problems for path table space allocation. Each volume of a volume set must include ‘a description of all the directories and files’ recorded on previous volumes as well as itself. This includes path table entries for those volumes, which clearly also limits the number of available entries in the path table, if a sub-volume contains a large number of directory resources.

Practical notes

Sorting. The second sorting criterion by parent directory location in the path table means that sorting must be performed piece-meal: the allocation of path table entries for directory level *n* require that the similar allocation for path table entries for level *n-1* has been finished. Path tables where this criterion has not been observed have been seen, though typically only in non-commercial software.

Some additional issues are treated in the following section.

Ideally, the consistency of all path tables within the path table pair as well as the directory structure should be verified before analysis starts.

Additional Issues

Standard Limits

ISO 9660 requires that a directory hierarchy rooted in a Primary or a Supplementary Volume Descriptor must not exceed 8 levels [ISO 9660, 6.8.2.1]. (The root directory is level 1, its immediate subdirectories are level 2, and so on.)

It also requires that length of a file path (directory names, path separators and final file identifier) must not exceed 255. (Path separators are not defined by ISO 9660: they are assumed to be what the execution platform of the receiving system specifies them to be.)

Thus, if an ISO 9660 volume contains non-conforming directory hierarchies or file paths, access to file or directory content may not be possible to ensure. This may affect questions of accessibility, which may consequently need access to a particular receiving system to decide. On most common computer platforms, there seems to be no problems with deeper-than-8 directory hierarchies, but this probably needs to be decided in a case by case basis.

Level of Interchange Limits

A related restriction is associated with Level 1 and Level 2 of Interchange, both of which limit files to consist of only one file section.

Imagine a file with multiple file sections, the first one of which contains innocuous information, and the second of which contains proscribed information. In an investigation concerning if person X had access to the proscribed information, it would become important to investigate how the receiving systems that person had access to would operate faced with such multi-segment files.

Duplicated Information

From the above notes on ISO 9660 file system structures, it is probably clear that there is a lot of duplicated information present.

The most widespread issue is that of logical block numbers, which are recorded in 32-bit both-byte format [ISO 9660, 7.3.3]. This provides a technical opportunity for the LSB half of the number to be different from the MSB half.

```
Volume:
  Root directory:
    File entry “FILE.TXT;1” → LBN [LSB half: 100; MSB half: 200]
```

The file FILE.TXT;1 is specified by a directory record in which the file location is self-inconsistent: one half of the both-byte-order field says the file is located at logical block number 100, and the other half logical block number 200. Depending on what the half the reading tool prefers to use, this will lead to different files extents being reached.

The logical block number for a directory is recorded in multiple places:

- in the directory itself as part of the ‘self’ directory entry,
- in each sub-directory, if any, as part of their ‘parent’ directory entry,
- in the parent directory, as part of the directory record for the directory
- in each path table recorded for the volume descriptor structure

and additionally the logical block number of the root directory is recorded in the relevant volume descriptor.

If, for example, a path table entry identifies one logical block number for the directory, while the parent directory record identifies another, implementations that rely on path table information for navigation will reach one directory, while implementations that go by directory content alone will reach another. (The possibility of having two path table groups allows for similar obfuscation, if the mandatory group identifies one directory address and the optional group identifies another.)

Additionally, if navigation from one directory to its parent involves referring to the ‘parent’ entry in the directory itself, this may also refer to a different directory than other paths do. In this situation we may have a situation where the paths

/DIR1/FILE.TXT, and

/DIR1/DIR2/../FILE.TXT, and even

/DIR1/./FILE.TXT

refer to different file content. In the second case, the ‘..’ path element is assumed to be equivalent to ‘follow the “parent” entry address to locate the subsequent directory’, rather than performing a path

normalization, followed by a path table look-up. In the third case, ‘.’ is assumed to use to the ‘self’ entry address in a similar manner, instead of being ignored. (Whether or not this actually happens in any particular case is obviously implementation dependent.)

We might even create a situation where the root directory does not identify itself as ‘self’ or ‘parent’ as stipulated by the standard and paths, but refer to some other directory, and we get a situation where paths such as

/../SECRET_DIRECTORY/FILE1.TXT

/./OTHER_SECRET_DIRECTORY/FILE2.TXT

may technically exist.

And, as file-hierarchy volume descriptors may be recorded multiple times, it may also be possible to have two primary volume descriptors, the first of which identifies one file hierarchy and the second one a different one. Which is actually used by a particular receiving system is clearly system dependent.

Practical Notes

Few if any current receiving systems will detect this kind of situation: minor informal tests suggest that forensic tools do not do better.

As such ‘’hidden’ or ‘obfuscated’ structures are present on the medium as logical sectors, their presence will be discoverable, either by searches for directory structures that are not accounted for in the ‘normal’ file trees, or by strict tracing of all references to directories, including ‘self’ and ‘parent’ addresses. Sector searches for keywords will also find them, although an analyst may not easily be able to identify if or how that sector is meant to be located.

For this reason, it may be enough to ensure that any examination of a ISO 9660 volume includes a search for unreferenced directory structures present, or by performing keyword searches at least once over the full ‘raw’ sector sequence. (Note: the sectors actually recorded, not the sectors/blocks the volume descriptor claims are present.) The ideal situation would of course be a full ISO 9660 validation test as a general test for file system structure soundness before more detailed examination is performed, but in the absence of tools that do this, it is unlikely to be a practical recommendation.

Content Category

ISO 9660 is not a ’live’ file system in that files and directories may be added or renamed at will. This makes many of the concerns of other file systems moot or irrelevant: there is no creation or deletion of files or directories, no renaming or moving of files or directories, no modification of file content. There are no structures for keeping track of free space.

File and directory allocation is done during what used to be called pre-mastering, but today is easier understood as the creation of the .ISO file. It is not modified later as far as an ISO 9600 file system is concerned; it can be modified from other viewpoints (such as multi-session CD/DVD), but those are not relevant here.

For this reason this section does not describe any of the details that *FSFA* typically describes. Instead it looks at those aspects of ISO 9660 interpretation that take place once all relevant file extents have been identified.

Access Controls

There is no access control of physical volumes: the first file structure level at which access controls are applied is at volume descriptor level.

Volume Descriptor Access Controls

The Level of Implementation used by a receiving system affects information access [ISO 9660, 13.5]. Level 1 allows the implementation to not make information in a Supplementary or Enhanced volume descriptor hierarchy available to a user.

Notes

Information about the level of implementation of the receiving system is not present on an ISO 9660 volume. It must presumably be collected from the actual receiving system itself. (This appears to lead to a situation where a 'non-forensic' image of a test CD, designed to identify if the receiving system would give access to known Supplementary or Enhanced volume descriptors, must be examined.)

It is not known if any receiving system implements this restriction. The possible existence of it must be known to an analyst, especially in cases where interpretation of access control is important.

The field Volume Effective Date and Time [ISO 9660, 8.4.29] specifies the date and time from which the information in the volume may be used. May be left undefined.

The field Volume Expiration Date and Time [ISO 9660, 8.4.28] specifies the date and time from which the information in the volume may be regarded as obsolete. May be left undefined.

(Note that these fields are relevant for Primary Volume Descriptors as well as Supplementary and Enhanced Volume Descriptors.)

Notes

The description is ambiguous in that 'regarded as obsolete' is not a clear parallel to 'may be used'. This may cause interpretation issues, or lead to lower levels of access control than intended.

There is no required form of access control: there is no requirement that users, for example, must be prevented from accessing information prior to effective date and time or after expiration date and time. This means that any detailed access control questions must be answered in the context of the actual receiving system.

Situations in which Effective Date and Time are specified as past Expiration Date and Time are not specified. A literal interpretation may block access entirely, whereas a pragmatic interpretation may result in less strict measures.

As files in one volume descriptor hierarchy may be referenced from later volumes in the same volume set, the question of interaction between source and destination access control also arises. As directory records refer directly to a logical block on a specified volume, it seems highly unlikely that such references involve referring to access control specified on the target volume, whether at the volume descriptor level or at any lower level. If target access control is not applied, such references may bypass access target control measures. There is no requirement that any access control in force for files on an earlier volume should be matched by later volumes in the same volume set.

File and Directory Access Controls

The File Flag 'Existence' [ISO 9660, 9.1.6, table 10] in the directory record indicates if the existence of the file should be made known to the user. If the flag is set to 0, the existence of the file shall not be made known, effectively hiding the file from the user.

The File Flag 'Protection' [ISO 9660, 9.1.6, table 10] in the directory record indicates if an extended attribute record containing additional access controls is present.

Notes

The standard text speaks only of the existence of the 'file'. It is not clear if this means that directory entries always must be made known to a user, or if it is only a short form for 'file or directory'.

If the File Flag 'Protection' is set, the Extended Attribute Record fields Owner Identification, Group Identification and Permissions are used for access control.

Owner Identification [ISO 9660, 9.5.1] specifies a 16-bit number that represents the owner of the file. The number 0 is special, and is interpreted as 'no owner'.

Group Identification [ISO 9660, 9.5.2] specifies a 16-bit number that represents the group of which the file owner is a member. The number 0 is special, and is interpreted as 'no owner', and requires that the Owner Identification field is also set to 0.

The field Permissions [ISO 9660, 9.5.3, table 13] specifies access controls for four types of users

1. System class users
2. The file owner specified by Owner Identification
3. The members of the group specified by Group Identification.
4. Other users

The specified file owner is required to be member of the specified group.

Each such access control includes read access (allowed or not allowed), and execute access (allowed or not allowed).

Notes

These access controls are not required by the standard to provide any protection settings different than default protection, although this would surely be the expected situation. If the Protection flag is set, the extended attribute record may be expected to express at least one non-default access controls.

Actual owner and group specifications are system dependent. Thus, any interpretation of this type of access control requires knowledge of how owner and group identifications are assigned and used by the receiving system.

There is no certain means of identifying that system. The subfield System Identifier [ISO 9660, 9.5.11] of the Extended Attribute Record is relevant only for interpretation of System Use subfields.

Additional Notes

The extent to which access control interpretation hinges on information not actually present in a volume may make these fields targets for obfuscation and analyst overload attacks by providing access control data that is not actually used, but where it is difficult or time consuming to show that such is the case.

Standard limits to numbers of directory levels, and length of file paths may affect access to non-standard volume files and directories (see section on Standard Limits).

File Data Character Set

An extended attribute record allows the data preparer to specify that the data in the corresponding file section should be interpreted as encoded by a character set defined in that record.

This situation differs from the similar one regarding character set for file and directory names in supplementary or enhanced volume descriptors. In that case only G0 and G1 characters can be specified, and control characters (C0 and C1) were not permitted. In the current situation, there is no such restriction.

This, in theory, allows a file section be recorded in Russian, Greek or any other supported character encoding. In practice, this is still subject to agreement between the originator and the recipient of the volume as to the character sets used for recording.

Note

This could be used for limited data hiding by using unregistered escape sequences (or even use of registered escape sequences with non-standard interpretation) to indicate encoded or encrypted contents.

As long the presence of such escape sequences is detected and examined, it cannot be done totally without trace, but would probably appear to be unusual but standard-conforming content. Decryption or decoding would in all likelihood still be beyond the capability of the average forensic analyst unless additional information is available.

Deciding if a receiving system allows specific file data character sets requires either assistance from the creator or the owner of that receiving system, or special test data volumes for those character sets.

Data Records

Use of extended attribute records allows ISO 9660 files to be organized in records: as a sequence of fixed-length records, or a sequence of variable-length records. These records may also be delimited in different ways, such as beginning with a LF character, and terminated by a CR character, or indicated by the first character of a line according to ISO 1539. The length of a record (or maximum length in case of variable-length) records can also be specified. [ISO 9660, 6.10, 9.5.8-10]

Unclear

It appears to be unspecified how data record specification interacts with file data character set specification: are record delimiters interpreted before character set decoding, or after?

Practical Notes

Very rare, possibly non-existent: no actual instances have been observed.

The only situation in which this type of data structuring would be used is probably when already existing databases needs to be placed on a CD without altering interpreting software on the receiving end. (Some older mainframe operating system such as VMS allowed record-structured files to be created. If such files were placed on ISO 9660 volumes, the record structure could probably be retained.)

Metadata Category

ISO 9660 metadata appears in volume descriptor records, in directory records, and also in extended attribute records.

This description only includes the most important fields. For full details see the standard text.

File-Tree Volume Descriptor Record Metadata

File-tree volume descriptors are those that refer to a directory structure, namely primary volume descriptors and supplementary or enhanced volume descriptors [ISO 9660, 8.4, 8.5]

System Identifier. If the System Area of the volume (*i.e.* logical sectors 0-15 left unspecified by ISO 9660 and left zero on most volumes) contains any information, the recipient system that is intended to interpret or act on that information should be specified in string format in the System Identifier field [ISO 9660, 8.4.5, 8.5.4]. However, this field is often used for other purposes – see Notes below.

Volume Identifier. The Volume Identifier field contains the name of the volume. As this is the main volume identification, it must be retained in all volume generations, but as already has been noted, this form of usage is rare in practice. [ISO 9660, 8.4.6, 8.5.5]

Volume Space Size. Identifies the number of logical blocks (!) in which the volume is recorded. [ISO 9660, 8.4.8]

Volume Set Identifier. Identifies the volume set of which this volume is a member. [ISO 9660, 8.4.19, 8.5.13]

Publisher Identifier.
DataPreparer Identifier. The identity of the publisher and the data preparer [ISO 9660, 4.4] of the volume group to which this volume belongs. May be empty.

Instead of containing the information directly, these fields may also specify the file in the root directory in which the information has been recorded [ISO 9660, 8.4.20-21, 8.5.14-15].

Application Identifier. Identifies the specification of how data is recorded in the volume group to which this volume belongs. May be empty.

Instead of containing the information directly, the field may also specify the file in the root directory in which the information can be found [ISO 9660, 8.4.22, 8.5.16].

Copyright File Identifier.
Abstract File Identifier.
Bibliographic File Identifier. Identifies a file in the root directory that contains copyright statement, contents abstract or bibliographic records for the volume or more. (See the standard text for exact interpretation.) May be empty. [ISO 9660, 8.4.23-25, 8.5.17-19]

Volume Creation Date and Time.
Volume Modification Date and Time.
Volume Expiration Date and Time.
Volume Effective Date and Time. Information when the volume was created and last modified, and the date and time when it becomes obsolete and becomes valid. May be undefined [ISO 9660, 8.4.26-29].

Practical Notes

System Identifier. Quite often the system area is left empty, but the system identifier field is used to record the information about the software that created single volumes. This may include CD creation software version.

In one known case, some versions of originating software add information that is not intended for any normal receiving system, but for reprocessing by the originating system itself.

Interpretation of this field must take into account that it is often used for purposes outside what the standard specifies.

Volume Identifier. The Volume Identifier field is usually reasonably distinctive, but in carelessly produced volumes it may be set to any default volume name of the burning software, such as 'DISC1', 'CD1', 'DVD1', 'CD_ROM', or to volume names based on date or time and so on. In some cases it may be left empty, in which case some recipient software may show information that is not actually recorded on the volume (such as 'Disk D:'), possibly leading to interpretation problems.

Directory Record Metadata

Recording Date and Time. The date and time at which the information in the extent was recorded [ISO 9660, 9.1.5]. May be undefined.

File Flags. The main use of this field is file-system interpretation related. It also allows the data preparer to indicate if a file is protected, and so indicate that additional protection- related information is present in the extended attribute record. Protection should not be observed by forensic tools, but if a particular receiving system is known to hide or otherwise prevent access to files and directories, it may be forensically significant.

Practical Notes

Recording Date and Time. Few ISO-creating programs set this exactly as defined; typically, the same date and time is recorded for all files, usually related to the time when recording started. Many current ISO-creating programs also allow the data preparer to set this field to any user-specified date, or to reflect the last modified timestamp of the original source files, which clearly is outside what the standard stipulates. Interpretation of these and other time stamps must take such non-standard usage into account.

***File Flags*.** As the protection-related setting requires the presence and correct interpretation of an extended attribute record, it is almost never found in the wild.

Extended Attribute Record Metadata

Extended attribute records provide additional attributes of files. This includes owner/group attributes and various permissions (related to the same attributes found for Unix file systems), and additional file time stamps.

Owner Identification.
Group Identification. 16-bit numbers that identify the owner and group for which permission

settings are defined. May be undefined [ISO 9660, 9.5.1, 9.5.2] ISO 9660 requires the specified owner to be a member of the specified group.

Permissions. 16-bit field defining the access rights of system-class users, of the owner, of members of the identified group, and of 'any user'. The individual and independent access rights are 'read' and 'execute'. [ISO 9660, 9.5.3]

File Creation Date and Time.
File Modification Date and Time.
File Expiration Date and Time.
File Effective Date and Time. Additional date and time stamps. Expiration date and time is the point in time after which the information in the file is no longer valid ('may be regarded as obsolete'), Effective date and time identifies the point in time before which the file information may not be used. Each and every field may be undefined [ISO 9660, 9.5.4-7]

Record Format.
Record Attributes.
Record Length. Defines any record structure present in the file that receiving information system is expected to handle. [ISO 9660, 9.5.8-9.5.10]

System Identifier.
System Use. System Identifier identifies a system that can act on the content of the System Use field both in the Extended Attribute Record and in the Directory Record that refers to the EAR. [ISO 9660, 9.5.11-12.]

Length of Application Use.
Application use. Allows up to 65535 bytes of application-specific data to be stored. [ISO 9660, 9.5.16]

Length of Escape Sequences.
Escape Sequences. Allow for specification of other character sets to be used for interpretation of file data. May be empty. [ISO 9660, 9.5.14, 9.5.18]

Unclear

Consistency. The only requirement placed on extended records for file sections of a multi-section file is that the record-related attributes must be the same. [ISO 9660, 9.6]

This appears to allow for different permission settings, different escape sequences/code sets as well as different extended time stamps for different sections of a file. Having different effective and expiration time stamps for various parts of a file, possibly with different user id and user access rights would seem to be permitted but very odd usage, for example.

Practical Notes

Extended attribute records are rarely found on modern ISO volumes.

Length of Extended Attribute Record. The length of an extended attribute record is defined by the number of logical blocks allocated to it and declared in the corresponding directory record [ISO 9660, 9.1.2]. This allows a maximum of 255 logical blocks to be used: with a logical block size of

2048 bytes, this means more than 500 kilobyte may be allocated. This field, however, records the number of bytes used: as it is recorded as a 16-bit value, only 31 blocks (31 * 2048 = 63488 bytes) may be accessed. Larger records (*i.e.* as specified in the directory record) may possibly hide information after the last used byte.

The length of an extended attribute record is also defined by the fields in the record [ISO 9660, 9.5]. This is 250 bytes of fixed allocated fields + up to 65535 bytes for application use + up to 255 bytes for escape sequences = 66040 bytes.

This appears to allow for around 450 kilobyte of data to be stored outside well-defined data structures per file section.

Time Stamp Metadata

ISO 9660 defines two types of time stamps, defined respectively in section 8.4.26.1 (for time stamps in volume descriptors and extended attribute records), and in section 9.1.5 (for time stamps recorded in directory records) [ISO 9660, 8.4.26.1, 9.1.5].

8.4.26.1-type time stamps include time information from years to hundredths of second, as well as offset from GMT time zone to a precision of 15-minute intervals (positive and negative). The range of years covered is from 0001 to 9999.

The information is stored as digits: years are stored as four bytes, each of which is expected to contain a decimal digit 0-9, *i.e.* (30)-(39) in normal ASCII encoding.

9.1.5-type time stamps contain similar information, but only to a precision of seconds, and are restricted to years in the range 1900-2155. In this format, year information is stored as a binary unsigned byte, indicating its offset from 1900, not as decimal digits.

Both time stamps formats reserve the ‘all-bytes-0x00’ field value to specify an explicitly undefined time stamp.

Unclear

The GMT time zone offset field is specified to be limited to -48-+52 15-minute periods, *i.e.* to −12:00 to +13:00 offsets.

However, since December 23, 1994, Line Island Time observes a GMT offset of +14:00, which thus would be formally illegal to use for an ISO 9660 system.

Tonga Summer Time zone also uses GMT offset of +14:00.

It is not known how this is handled is real life, especially as some operating system also appear to lack (or have lacked) time zone support that include Line Island Time.

Practical Notes

As time stamps consist of decimal digits in byte fields, it is technically possible to have time stamps designating illegal dates and time.

In some situations, reading applications have been observed to mistranslate such dates into legal dates. For example, a date specified as (digit-by-digit) 2000-02-33 has been seen translated into 2000-03-04. This may be convenient in end-user applications, but it can hardly be considered suitable in a forensic analysis setting.

File and Directory Identifier Category

ISO 9660 has different rules for directory tree identifiers, both as regards to type (file identifiers have different rules than directory identifiers), and as regards to the type of the volume descriptor hierarchy they are part of.

Primary and Supplementary Volume Descriptor Hierarchies

Directory identifiers in primary volume descriptor hierarchies may consist of 1-31 d characters. (d characters are the characters 0-9, A-Z and _ = underline [ISO 9660, 7.4.1]). There are two exceptions. The special directory entry for the current directory is given the identifier (00), while the entry identifying the parent directory is given the identifier (01). These identifiers can only be assigned by the originating system when the ISO 9660 file system is created, and they have a purely internal purpose, *i.e.* the receiving system should not allow a user to refer to them by those identifiers.

In a supplementary volume descriptor hierarchy, the allowable characters are referred to as d1 characters, and are decided by agreement between the originator and the recipient. Thus, in a supplementary directory hierarchy, there is *no* explicit or implicit rule about what characters may be used, unless the details of that agreement are known.

File identifiers in primary and supplementary volume descriptor hierarchies consist of three parts: the file name, the file name extension, and the file version number. File names and file name extension may only consist of d or d1 characters (*i.e.* the same as for directory identifiers). Either the file name or the file name extension may be empty, but not both at the same time. The length of the file name and file name extension together may not exceed 30.

The file name and the file name extension are separated by a SEPARATOR1 character, and the file name extension and the file version are separated by SEPARATOR2. These are mandatory: they cannot be omitted. SEPARATOR1 is the byte (2E), *i.e.* a period in ASCII-based encodings, and SEPARATOR2 the byte (3B), *i.e.* a semicolon [ISO 9660, 7.4.3].

The file version consists of digits representing a number from 1 to 32767.

Examples

`name.ext;1`	(legal)
`name.;1`	(legal – file identifier without file extension)
`.ext;1`	(legal – file identifier without file name)
`.;1`	(illegal – file name and file extension may not both be empty)
`file.tar.gz;1`	(illegal – only one of each SEPARATOR may be present)

File version numbers are used in multi-volume sets to help identify file versions. As all earlier content of a multi-volume set directory must also be present in the current volume, the presence of modified files can be identified, and the location of them can easily be established.

Enhanced Volume Descriptor Hierarchies

File and Directory identifiers can be up to 207 bytes long, but allowed characters are agreed on between originator and recipient.

Furthermore, no SEPARATOR characters are defined for these file identifiers, and no file name extension or file version is present in them.

Unclear points

File names. File names in an enhanced volume descriptor hierarchy seem oddly defined:

> *Within a Directory Hierarchy that is identified in an Enhanced Volume Descriptor, a File Identifier shall not be specified as certain character sequences. These sequences shall be subject to agreement between the originator and recipient of the volume.*

This does not appear to make good sense as it stands, and an implementer will almost certainly need to interpret this in one way or another. (Does the requirement try to say that certain character sequences, agreed on by originator and recipient, must *not* be part of a file identifier? Such as 'PRN' or 'NUL' on FAT or NTFS?)

File version number. The standard text says that the file version number consists of "digits representing a number from 1 to 32767."

This is somewhat ambiguous as regards to leading zeroes: what number do the digits "01" represent? Is it different from the file version number "1"? Or is it an incorrect file version number?

The requirement that a file identifier in an enhanced volume descriptor hierarchy does not contain any file version probably changes the behaviour stipulated elsewhere in the standard, namely that all versions of a file on a multiple-volume set shall be present in the directory structure. With the current definition, this requirement cannot clearly be provided. If there are no file version numbers at all, some unexpected behaviour as concerns file identifier sorting may perhaps be observed, if a file identifier contains a portion that looks like a file version number but actually is not one.

Practical points

File identifiers. Some originating systems allow file identifiers that do not follow the ISO 9660 rules, and, for example, do not include a file version or omit SEPARATOR2. Such non-standard file identifiers must be verified to be handled correctly by the forensic tools used for analysis.

Application Category

While the standard provides for both system- and application-specific data to be provided at many levels of the volume and file tree hierarchies, there is no known interpretation within the standard itself.

Some known additional standards (such as El Torito) or possibly non-conforming extensions (such as Joliet) are indicated in section Related Standards on p. 37.

ISO 9660 Data Structures

File System Forensic Analysis provides tables describing the allocation of fields in important file system structures.

Initially, it was planned to do the same thing here, but as all technical details are available in both the ISO 9660 standard as well as in the free ECMA 119, rev. 3 documents, the need for repeating the information seems minimal, especially the information about subfield layout and the risk for errors in such repetition is considered unacceptable.

Instead, this chapter provides references to the standard with occasional comments where the standard may not be sufficiently clear for interpretation

Volume Descriptors

[ISO 9660, 8, 6.7]

Volume Partition Descriptor

[ISO 9660, 6.4.7, 8.6]

The standard allows space within the data area of the volume to be assigned to one or more volume partitions. It requires such partitions to be allocated on logical block boundaries. This suggests that file and directory content is the primary area of application.

While this volume descriptor type is documented in the standard, there is no standard information about how it should be interpreted or what action should be taken when one is identified. Thus, the interpretation of a volume partition descriptor is entirely left to the recipient system identified by it, and as there is no register of such systems, volume partitions are white areas on the ISO 9660 map.

Practical Notes

No volume partitions have been seen on ordinary CD volumes (software distribution, games, etc.).

As this is an unknown system-specific structure, any presence of volume partition descriptor records in a ISO 9660 volume seems to be interesting enough to flag up for a forensic analyst, as its presence may suggest that additional processing or information is required for correct interpretation of the volume.

A possible use of volume partitions may be to support additional forms of access control or some form of encryption of the data contained in the partition.

Directory Records

[ISO 9660, 9, 6.8]

Practical notes

Extended Attribute Record Length. Section 9.1.2 does not clearly state in what units Extended Attribute Record Length should be recorded in a directory record, except that it should be recorded according to 7.1.1, *i.e.* an 8-bit quantity.

At the same time 9.5.10 states that the Record Length of an extended attribute record should record the number of bytes of the record according 7.2.3, *i.e.* a 16-bit quantity.

The intended use is almost certainly that 9.1.2 refers to ‘number of logical blocks’. But as the standard is not clear, this may be an area where differing interpretation may arise.

Data Length. The data length of a directory is fairly often recorded as the sum of the lengths of the directory records contained in the directory. This is incorrect: the data length of a directory must always be a multiple of the logical sector size.

Carving. As directories are required to start with two special directory records, which use the reserved file identifiers (00) and (01), it is reasonably easy to identify directory structures without going by the directory tree or the path tables:

> Look at logical sector boundaries only
>
> Look for the byte pattern of the two special directory records first in that sector.

If found, the (00) entry is expected to contains the data length of the directory, and so identify the sectors that have been allocated for it. (ISO 9660 does not allow directories to be interleaved.)

If the current logical block matches the criteria above, but the current logical block number does not match the number in the (00) entry, there may be some form of obfuscation going on: the ‘current directory’ entry points elsewhere.

Path Table

[ISO 9660, 6.9, 9.4]

Extended Attribute Record

[ISO 9660, 9.5, 9.6]

File Data Storage

File Extents [ISO 9660, 6.4]

File Sections [ISO 9660, 6.5]

File Record structures [ISO 9660, 6.10]

File Character Set Coding [ISO 9660, 9.5.18]

ISO 9660 Conformance

This section is included as it may help interpretation of some non-standard and borderline usage. It also touches of the issue of 8+3 as a maximum file identifier length sometimes said to be stipulated by the standard.

ISO 9660 states requirements for three areas of conformance: the CD-ROM, the originating system, and the receiving system.

The term “CD-ROM” is used also in the latest versions of the standard, even though DVD and Blu-ray Discs are common, and CD-R and other recordable optical media are available. As ISO 9660 is formulated in a way that makes it independent of the underlying recording standard (CD-ROM, CD-R, CD-RW, etc. – see section ‘Overview: Logical Sectors and Logical Blocks’ above) we will take this term as synonym to “CD-ROM or other suitable medium” for typical forensic analysis.

The “originating system” refers to the information system that is used to create the file system. Today, this is typically end-user ISO burning software, producing an .ISO image or equivalent. Around 1990, it would have been a service provided by a company that did CD-ROM pressing, or was closely related with such company.

The “receiving system” today is typically the system software that mounts and accesses ISO-formatted media on personal computers. It may also refer to software that implements any system- or application-specific data access, as identified by the ISO 9660 volume itself. In very early descriptions of what would become the ISO standard, special reading applications were sometimes hinted at as possibilities, possibly influenced by the concept of hypertext as described by Ted Nelson.

The criteria for CD-ROM conformance are, basically, the requirements stated in Section II of the standard [ISO 9660, 6–10] while the standards for the originating and receiving systems are stated in Section III [ISO 9660, 11–13].

Note

The fact that the term “CD-ROM” has not changed in the standard since 1988 is difficult to interpret. As CD-ROMs are pressed in special plants, and typically are produced in minimum batch sizes of 300-500 copies, this might be taken to suggest that such mass-produced media (whether CD-ROM, DVD-ROM or BD-ROM) is the intended area of use for ISO 9660.

The alternative that “CD-ROM” is the only medium on which an ISO 9660 file system may be recorded (*i.e.* to the exclusion of other read-only or recordable optical media) is possible, but does not seem entirely reasonable to base a forensic analysis on.

CD-ROM

In chapter 10, three levels of interchange are formulated, and summarized below:

> Level 1: No multi-extent files, and in primary and supplementary volume descriptor hierarchies only 8+3 file identifiers, and 8 character long directory identifiers

Level 2: No multi-extent files

Level 3: No restrictions

Chapter 10 only specifies the levels of interchange: it does not state any requirements. The requirements that bear on these are found in chapter 2, where the standard requires an ISO 9660 volume to be accompanied by a statement of conformance that identifies the lowest level of interchange to which the volume conforms [ISO 9660, 2.1], and in chapter 12, where the standard requires the originating system to record files and file information according to one of the interchange levels.

(It may be worth noting here that Level 1 does require support of associated files, as well as extended attribute records.)

Note

The assumption or interpretation that ISO 9660 (as a standard) requires file identifiers to be at most 8+3 characters length is thus probably a misinterpretation. A data publisher who wishes to provide volumes for users whose receiving system only can handle Level 1 CD-ROMs is however forced to use an originating system that can create volumes that follow Level 1.

Originating system

The requirements on the originating systems are stated in Chapter 12.

The requirements basically state that the “data preparer” (a user role, involving the technical preparation of an ISO 9660 volume) shall be allowed to decide various information and parameters.

Some of these are self-evident such as the volume identifier, copyright file identifier, etc.

Some are less obvious, such as the logical block size, location of path tables as well as location of individual files and directories, and the content of the system area of the volume, *i.e.* logical sectors 0-15 that are left unspecified by the standard.

Few if any modern ISO burning software claiming ISO 9660 conformance actually allow a user to specify this type of information.

Some information is notable by its absence: the data preparer is *not* explicitly allowed to change the Date and Time of Recording, something several modern ISO burning software products actually do allow the user to do.

It is also worth observing that none of these requirements state that the data preparer shall be able to specify that a volume of interchange level 1 (see above) shall be produced. This decision is presumably made at a different level.

Note

Knowledge of the point on which some particular originating software does *not* conform to ISO 9660 requirements may be important for correct interpretation of evidence. For example, assuming that date and time of recording is recorded as stated by the standard is not in general sound practice.

Receiving system

The requirements on the receiving systems are stated in Chapter 13.

These requirements are basically what information the receiving system must make available to a user, as well as what information the receiving system may withhold from a user. As conformance to these requirements typically is not important to a tool for forensic analysis, which should provide all information, no further details will be provided here, but the ability to identify information that may be withheld may be important for interpretation issues.

It may, however, be important to ensure that a forensic tool does not try to withhold any of the information that ISO 9660 may allow it, as a receiving system, to withhold.

Receiving systems are classified in two levels of implementation [ISO 9660, 13.5]:

> Level 1: File data and metadata in supplementary or enhanced volume descriptors may be withheld from the user.
>
> Level 2: No restrictions

Practical Notes

Associated Files. No receiving system is required to make associated files available to the user. [ISO 9660, 13.1] Interpretations based on associated files, or their contents, must thus be based on what the actual receiving system allows.

Additional Topics

File System Interpretation

Most, possibly all, modern originating systems (*i.e.* ISO 9660-burning software) as well as recipient system (disc mounting software) treats file-related volume descriptors as independent entities: when a file hierarchy is written to a CD, the files are listed in the primary volume descriptor (possibly after name adjustment to fit a real or imagined 8+3 file name restriction, and file identifier character set conversion to fit the restricted ISO 646 code table), and usually also in a Joliet volume descriptor. When that CD is mounted by a recipient system, only one of these file descriptors are made available to the user.

Is this the ‘right’ and ‘correct’ interpretation: volume descriptors are independent and if one volume descriptor is selected, it is to the exclusion of any other volume descriptor that may be present?

Another possible interpretation could be that no file data or metadata may be excluded (except as far as ‘hidden’ files [ISO 9660, 9.1.6, ‘Existence’] or file protection settings [ISO 9660, 9.5.3] are concerned), and that the file tree presented to the user should be the ‘sum’ of all file trees in a volume.

This interpretation has no explicit support from the standard, and comes with some precedence issues (if all volume descriptor file trees contain the file ‘README.TXT;1’ in their root directories, which should be shown to the user?), although some of those may be viewed as system or application-specific decision, and so solved elsewhere, rather than standard-related questions.

The question may be important for data providers: can a file in one volume descriptor file hierarchy ‘assume’ that a file in another volume descriptor files hierarchy is guaranteed to be accessible? This type of question may be important for web-structured content, where one node contains a hyperlink to a file present in another file hierarchy.

While it is unsafe to argue from absence of evidence or standard requirements, it may be possible that the absence of them in this case may point to the interpretation that ISO 9660 volumes were not intended for general use, but only for specific customers who shared the same recipient software.

It seems unlikely that this particular question will become important in a forensic case, but if it does, it will probably require additional evidence as support for a particular interpretation.

Related Standards

High Sierra

This is a precursor to ISO 9660.

Microsoft announced support for the High Sierra file system in 1986 for MS-DOS 3.1 or later [*ComputerWorld*, September 22, 1986, p. 13]. (The same issue of *ComputerWorld* announced that 3M had produced the first CD-ROM to High Sierra standards. The volume contained census data and business pattern statistics.)

Some degree of support for High Sierra volumes was provided by Microsoft MSCDEX utility, used in DOS and early Windows releases.

No authoritative source is known.

Note: While not the same format as ISO 9660, there are strong similarities, and High Sierra volumes may be tentatively identified by having the string 'CDROM' at approximately the same location of volume descriptors of ISO 9660 have the string 'CD001'.

El Torito

Specified in

> Phoenix Technologies and IBM: *"El Torito" Bootable CD-ROM Format Specification* version 1.0 (January 25, 1995)

This is a specification of a system that reads and interprets some ISO 9660 boot records. Its main forensic interest is that the presence of boot code can be identified and the boot code and boot options may be analyzed.

A drawback is that it requires boot records to be located at a specific logical sector (17), while ISO 9660 allows them to be anywhere in the volume descriptor set.

The text is also incorrect in places: in one case, for examples, CD sectors are said to be 800 bytes, instead of 2048 bytes. (This is probably a misprint for 800_{16} or 0x800 bytes)

No authoritative download from Phoenix or IBM has been identified.

https://www.intel.com/content/dam/support/us/en/documents/motherboards/desktop/sb/specscdrom.pdf

A short paper on some problems with the standard or its implementations:

> Andrew Smith: *EL Torito Specification Supplement* (22 September 2006)
> (web document, retrieved on 2020-09-01 from http://littlesvr.ca/isomaster/eltoritosuppl.php)

TRANS.TBL

This is a non-ISO 9660 extension in that is not labelled as a System or Application required by the receiving system.

It addresses the limit of 8+3 character file identifiers and 8 character directory identifiers (see section ISO 9660 Conformance above for more info) by placing a file named TRANS.TBL (? See notes below) in every relevant directory of the file trees, specifying the translation between the actual file or directory identifier on the CD to a 'long' identifier.

For additional details, see

> https://en.wikipedia.org/wiki/TRANS.TBL

Notes:

No specification has been found. Known to have been in use at least from 1995, but is probably older.

Details about translation file name vary: some say 'TRANS.TBL' others 'TRANSL.TBL;1'. (If version number is a specified part of the file identified, volume sets with changes in this file may cause interpretation issues.)

TRANS.TBL files have been observed in Joliet volume descriptor file trees: but as Joliet allows up to 64 characters in file and directory identifiers, the need for them is not clear, and they may be ignored by receiving system.

And of course, the presence of TRANS.TBL files does not mean that the receiving system can or will interpret them correctly.

Use of TRANS.TBL might conceivably fall under the kind of uses to which Volume Partitions may be put. No such use is known, however.

Apple ISO 9660 Extensions

These extensions provide HFS properties (ProDOS properties are also mentioned in the body of the specification).

It uses ISO 9660 directory record system use field in manner similar to, but not compatible with Rock Ridge SUSP.

Specification: Apple Technical Note FL36 : Apple Extensions to ISO 9660

> http://developer.apple.com/technotes/fl/fl_36.html

This document does not appear to be available any more, but a copy can be found at Internet Archive Wayback Machine

> https://web.archive.org/web/20081226015418/http://developer.apple.com/technotes/fl/fl_36.html

Rock Ridge Extensions

These extensions are often referred to as IEEE P1281 and IEEE P1282, which gives the impression that they are (still) preliminary IEEE standards. As this is incorrect, those particular designations are misleading, and should be avoided.

Rock Ridge System Use Sharing Protocol (SUSP)

This extension defines a framework for ISO 9660 directory record System Use fields in a recipient-independent manner.

IEEE P1281: System Use Sharing protocol (Draft Standard Version 1.12) used to be available from

http://www.ymi.com/ymi/sites/default/files/pdf/Systems%20Use%20P1281.pdf

but can now only be retrieved through the Internet Archive Wayback Machine.

Notes:

Continuation areas extend the space available for SUSP use, are allocated in logical blocks (presumably – it is not explicitly stated) and are identified by their first four bytes: 'C' 'E' (1C 01). (Later version specification may have different content. The 'C' and 'E' bytes are not fully specified as the standard does not specify what character code is used.) This may be used to locate continuation areas if normal file tree parsing is not possible, for example if the recording medium has been damaged.

A strict application of ISO 9660 probably requires that this use of the System Use field is labelled by providing a corresponding System Identifier (not defined by SUSP) in an associated Extended Attribute Record [ISO 9660, 9.5.11]. However, this imposes a cost of one logical block per directory record that uses SUSP, which seems excessive: if an extended attribute record was required, it would be easier to use it for SUSP-related data, instead of having to deal with continuation areas.

Rock Ridge Interchange Protocol (RRIP)

This extension defines a method of recording POSIX file information, using the SUSP framework (see above).

As far as ISO 9660 interpretation goes, RRIP allows specification of alternate file name, POSIX time stamps, and sparse file data.

Some of the information specified by this protocol may also be present in extended attribute records: there are no clear requirements for consistency between RRIP information and extended attribute record information.

IEEE P1282: Rock Ridge Interchange Protocol (Draft Standard Version 1.12) used to be available from

http://www.ymi.com/ymi/sites/default/files/pdf/Rockridge.pdf

but can now only be retrieved through the Internet Archive Wayback Machine.

Microsoft Joliet

See first chapter, and particularly the information related to ISO 9660:1988 / Amd. 2 for source information.

A summary of relevant chances are printed in the standard [ISO 9660, B.2].

New Mandatory Character Set: ISO 10646:1993 / UCS-2

Three coding systems of ISO 10646 that *must* be supported by the recipient are introduced. The coding systems and their corresponding escape sequences are

ISO/IEC 10646:1993, UCS-2, Level 1	(25 2F 40)
ISO/IEC 10646:1993, UCS-2, Level 2	(25 2F 43)
ISO/IEC 10646:1993, UCS-2, Level 3	(25 2F 45)

UCS-2 is a coding system for ISO 10646:1993 character set that is not supported by ISO 2022, as is formally required for ISO 9660. The escape sequences are registered, and can thus be used as indicators that the volume descriptor hierarchy follows the Joliet specification, but a 'pure' ISO 9660 reader might not accept or interpret them. (It is not clear from the Joliet specification if they can also be used to specify file data encoding, or if they may only be used for file/directory identifiers and metadata.)

UCS-2 is now obsolete, and it is no longer described as part of ISO 10646: for a full description, the referenced standard or a version from approximately the same time is required (ISO 10646:2003 does contain a description, for example, but details may have changed since the 1993 version).

Very roughly, UCS-2 was a pure 16-bit encoding system for the Unicode BMP (Basic Multilingual Plane). It was *not* the same as any form of UTF-16, which reserves the range U+D800 – U+DFFF for surrogate pairs in order to represent characters outside BMP, such as emoji glyphs. UCS-2 did not reserve any characters for such use, and so could only encode characters in the range U+0000 – U+FFFF.

UCS-2 Level 1 restricted the encoded data to not contain characters listed in section B.1 of the ISO 10646:1993 standard. (These are mainly various combining characters, *i.e.* characters that are intended to be combined with a base character to form a combined glyph, such as a diaeresis accent combined with the letter a to form an ä.)

Level 2 restricted the data to not contain characters listed in section B.2 of the standard. (These are a subset of the characters excluded by Level 1.)

Level 3 did not impose any restrictions.

Joliet requires characters to be recorded in MSB format, *i.e.* most significant byte first. (UCS-2 did not have any such requirements.)

Practical Notice

CDs with the Joliet volume descriptor listing more than one of the UCS-2 escape sequences have been observed. This is technically impossible: only one coding scheme can be active for a certain byte/character. This may require special interpretation to ensure affected characters are correctly interpreted.

Files in recent Joliet volume descriptor trees have been observed to contain Unicode characters that are outside UCS-2 code table (*e.g.* \U+1F608_😈_.TXT). This may indicate that there are later releases of the Joliet specification; it may also indicate that some recording systems use the UTF-16 coding system instead of UCS-2.

File and Directory Names

There are no d1 characters identified for Joliet: any character in the supported character set can be used with the exception of control characters (U+0000 – U+001F), and the characters '*', '/', ':', ';', '?' and '\'.

Note: this allows the use of SEPARATOR 1 in Directory Identifier (*i.e.* 'DIRECTORY.EXTENSION' is a legal directory identifier.) If the receiving platform uses other characters for path separators, the presence of those characters in ISO 9660 file names may cause interpretation problems.

The length of a Directory Identifier or the combined length of File Name and File Name Extension may not exceed 128 bytes = 64 UCS-2 characters.

String data

FILLER is defined to be (00).

ISO-9660:1999

Despite the official-looking name, this is *not* a version of the ISO 9660 standard, nor has it been one. It might have been a draft ISO/IEC standard (in which case the correct designation should have been ISO/IEC DIS 9660:1999), but it is not listed as a past draft standard by ISO.

Currently, the best guess is that it may have been an attempt to create a draft standard that for some reason failed.

Note

The following document (of unknown provenance) may be a draft:

> https://pismotec.com/cfs/iso9660-1999.html

Annex B of this document contains notes on historical background, which may suggest that some or all of the changes introduced in ISO 9660:1988/Amd. 1 were developed as part of this effort. The Japanese standard JIS X 0606:1998, cited in the document but apparently available only in Japanese, may have more information.

Important Changes

Corrected the use of ‘logical block address’ and the abbreviation LBA to ‘logical block number’ and LBN, in accordance with the standard.

Collected all access control-related information in a single place (ISO 9660 File System Analysis / Content Category).

Added notes on restrictions of file paths and directory structure depths as having possible impact on questions of data accessibility.